Proceedings

TWENTY-FIFTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

September 30-October 2, 1987

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

621.3811
Al5
1987 v.2

PROCEEDINGS

TWENTY-FIFTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

[redacted]ME I

P. R. Kumar
Michael C. Loui
Conference Co-Chairmen

Conference held
September 30-October 2, 1987
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

Dedicated to

Mac E. Van Valkenburg

621.3811
Al5
1987:2

Engin

FOREWORD

The Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on September 30 through October 2, 1987. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain the papers presented at the Conference.

We are grateful to Professor W. M. Wonham of the University of Toronto for his plenary lecture "Logic and Language in Control Theory." We are also grateful to Professor B. H. Krogh of Carnegie-Mellon University for organizing the invited session "Discrete Event Systems" and to Professor M. W. Spong for organizing the invited session on "Robotics." We would like to thank all the session chairs and the authors for their contributions.

We appreciate the support of Professor M. E. Van Valkenburg, Dean of the College of Engineering and founder of the series of Allerton Conferences, Professor T. N. Trick, Head of the Department of Electrical and Computer Engineering, and Professor W. K. Jenkins, Director of the Coordinated Science Laboratory. We would also like to thank our colleagues Professors J. A. Abraham, E. Arikan, K. S. Arun, P. Banerjee, M. T. Basar, Y. Bresler, D. J. Brown, W. K. Fuchs, B. E. Hajek, T. S. Huang, R. Iyer, P. V. Kokotovic, H. Lee, J. V. Medanic, D. C. Munson, J. H. Patel, W. R. Perkins, K. R. Poolla, H. V. Poor, F. P. Preparata, M. B. Pursley, D. V. Sarwate, M. W. Spong, and B. W. Wah for their generous assistance in reviewing papers and organizing the papers into sessions.

We are grateful to Mr. Steven Williams for his handling of the business affairs of the Conference, to Mr. Robert Ebeling and his assistants for their timely printing of large numbers of programs and registration packets, and to Ms. Rose Harris, Ms. Tina Lipscomb, Ms. Dixie Murphy, and Ms. Janet Reese for their capable assistance in handling the Conference announcement, registration, program preparation, and editing and sales of these *Proceedings*.

P. R. Kumar

Michael C. Loui

TABLE OF CONTENTS

FOREWORD i

PLENARY LECTURE

LOGIC AND LANGUAGE IN CONTROL THEORY 1
W.M. Wonham
University of Toronto

I-A: COMBINATORIAL ALGORITHMS

Chairman: F.P. Preparata
University of Illinois at Urbana-Champaign

COMPLEXITY RESULTS FOR VERTEX-DELETION BIPARTIZATION AND VIA MINIMIZATION PROBLEMS 4
H-A. Choi, K. Nakajima, and C.S. Rim

ALGORITHMS FOR REGULAR PROPERTIES IN RECURSIVE GRAPHS 14
S. Mahajan and J.G. Peters

AN OPTIMUM $\theta(n \log n)$ ALGORITHM FOR FINDING A CANONICAL HAMILTONIAN PATH AND A CANONICAL HAMILTONIAN CIRCUIT IN A SET OF INTERVALS 24
G.K. Manacher and T.A. Mankus

COMPUTING A CONDITIONAL EDGE-CONNECTIVITY OF A GRAPH 34
A-H. Esfahanian and S.L. Hakimi

SMALL-LARGE DYNAMIC PROGRAMMING FOR TREES 36
J. McHugh and Y. Perl

APPROXIMATE STRING MATCHING WITH CONTEXT-DEPENDENCY 46
A.A. Bertossi, E. Lodi, F. Luccio, and L. Pagli

DETERMINISTIC MOVE-TO-REAR LIST ORGANIZING STRATEGIES WITH OPTIMAL AND EXPEDIENT PROPERTIES 54
B.J. Oommen, E.R. Hansen, and J.I. Munro

AN INVESTIGATION OF SELF-ORGANIZING HEURISTICS FOR DOUBLY LINKED LISTS 64
M.E. Bellow

HOW WELL ARE BINARY SEARCH TREES BALANCED? 66
W. Szpankowski and V. Rego

I-B: ROBUST AND NONLINEAR CONTROL

Chairman: R. Ortega
National University of Mexico

NECESSARY AND SUFFICIENT CONDITIONS FOR STABILITY OF POLYTOPES OF MATRICES THROUGH TESTS OF LOWER DIMENSIONAL SUBSETS 72
C.L. DeMarco

SLIDING REGIMES IN ANALOG SIGNAL ENCODING AND DELTA MODULATION CIRCUITS 78
H. Sira-Ramirez

FEEDBACK STABILIZATION OF UNCERTAIN SYSTEMS 88
P.P. Khargonekar, I.R. Petersen, and K. Zhou

OBSERVER-BASED DESIGN OF UNCERTAIN SYSTEMS: RECOVERING STATE FEEDBACK ROBUSTNESS UNDER MATCHING CONDITIONS 97
F. Esfandiari and H.K. Khalil

CONTROL OF A PM STEPPER MOTOR USING NONLINEAR OBSERVERS AND CONTROLLERS 107
M. Zribi and J. Chiasson

KHARITONOV'S THEOREM: IS THERE A GENERALIZATION FOR DEPENDENT COEFFICIENT VARIATIONS? 117
B.R. Barmish

EIGENVALUE BOUNDS FOR THE SOLUTION OF DISCRETE STOCHASTIC RICCATI EQUATION 119
E. Yaz

ROBUST FLIGHT CONTROL SYSTEM DESIGN TECHNIQUE USING GAIN SPACE WITH PERFORMANCE AND HANDLING QUALITIES CONSTRAINTS 121
G. Aziz and J. Ho

I-C: DETECTION

Chairman: W.L. Root
University of Michigan

SOME RESULTS ON THE ROLE OF 2-ALTERNATING UPPER PROBABILITIES IN SIGNAL-PLUS-NOISE MODELS 123
A. Sadrolhefazi and T.L. Fine

BOUNDS ON THE VARIANCES OF IMPORTANCE SAMPLING SIMULATIONS IN DIGITAL COMMUNICATION SYSTEMS 125
D. Lu and K. Yao

ON THE APPLICATION OF IMPORTANCE SAMPLING TO THE ANALYSIS OF DETECTION SYSTEMS 135
G. Orsak and B. Aazhang

WALD'S IDENTITY AND SEQUENTIAL TEST PERFORMANCE WITH DEPENDENT DATA 145
J.S. Sadowsky

DISTRIBUTED DETECTION WITH CORRELATED SENSOR NOISE 155
R. Viswanathan, V. Aalo, and S.C.A. Thomopoulos

A DIFFERENTIAL GEOMETRIC APPROACH TOWARD ROBUST SIGNAL DETECTION IN NONSTATIONARY AND/OR DEPENDENT NOISE 163
M.W. Thompson and D.R. Halverson

SIGNAL DETECTION IN ARCTIC UNDER-ICE NOISE 172
P.A. Nielsen and J. Thomas

I-D: DIGITAL SIGNAL PROCESSING

Chairman: K.S. Arun
University of Illinois at Urbana-Champaign

DETECTION OF SPEECH SIGNALS AT THE OUTPUT OF A COCHLEAR MODEL 182
B. Monderer and A.A. Lazar

LIMIT CYCLES IN LINEAR SHIFT VARYING DIGITAL FILTERS 192
T. Bose and D.P. Brown

BEST ORDERINGS FOR FLOATING POINT ADDITION 199
T.G. Robertazzi and S.C. Schwartz

A RATIONAL VIEW OF THE FAST FOURIER TRANSFORM 201
C.M. Fiduccia

SPLIT LEVINSON ALGORITHM FOR TOEPLITZ MATRICES OF ANY RANK PROOFILE 203
M.K. Ciliz and H. Krishna

AN ALGORITHM FOR ESTIMATING THE FREQUENCIES AND AMPLITUDE MATRICES CORRESPONDING TO MULTIDIMENSIONAL HARMONIC SIGNALS IN NOISE 205
C. Foias, A.E. Frazho, and P.J. Sherman

ON THE OPTIMUM DELAY IN FREQUENCY-SAMPLING ALE 215
S.M. Kuo and M.A. Rodriguez

SYNTHESIS AND DESIGN OF STRUCTURALLY PASSIVE FULLY RECURSIVE HALF-PLANE 2-D DIGITAL FILTERS 217
S. Basu and K. Mahtre

I-E: PARALLEL PROCESSING

Chairman: B. Wah
University of Illinois at Urbana-Champaign

CONCURRENT PROCESSING WITH RESULT SHARING: AN EFFICIENT COMPUTING PARADIGM 227
S. Krishnaprasad

PARALLELIZING 'LEAST-COST' BRANCH-AND-BOUND ALGORITHMS 236
G. Lee

THE SIMPLE SIMON PROGRAMMING ENVIRONMENT: A STATUS REPORT 238
J.E. Cuny, D.A. Bailey, J.W. Hagerman, and A.A. Hough

THE LATTICE STRUCTURE OF A BUS ORIENTED MULTIPROCESSOR MODEL 248
A.A. Lazar and T.G. Robertazzi

MIXED-MODE COMPUTING WITH THE PASM SYSTEM PROTOTYPE 258
S.A. Fineberg, T.L. Casavant, and T. Schwederski

ONE-TO-K COMMUNICATION IN DISTRIBUTED-MEMORY MULTIPROCESSORS 268
H-A. Choi, A-H. Esfahanian, and L.M. Ni

II-A: DISCRETE EVENT SYSTEMS

Chairman: B.H. Krogh
Carnegie Mellon University

ON SYNCHRONIC-DISTANCE AND B-FAIR RELATIONS FOR PETRI NETS 270
M. Silva and T. Murata

INFINITESIMAL PERTURBATION ANALYSIS OF A MULTICLASS ROUTING PROBLEM 279
P. Vakili and Y-C. Ho

AUGMENTED CHAIN ANALYSIS OF MARKOV AND SEMI-MARKOV PROCESSES 287
S.G. Strickland and C.G. Cassandras

WEAK CONVERGENCE OF SAMPLE PATH DERIVATIVES FOR THE WAITING TIME IN A SINGLE SERVER QUEUE 297
M.A. Zazanis

CONSTRUCTION OF A T-BASE AND DESIGN OF A PERIODIC FIRING SEQUENCE OF A LIVE AND BOUNDED PETRI NET 305
Q.W. Ge, T. Tanida, N. Ono, and K. Onaga

T-COMPLEMENT PETRI NETS 315
R.S. Sreenivas

CONTROLLED PETRI NETS AND MAXIMALLY PERMISSIVE FEEDBACK LOGIC 317
B.H. Krogh

SUPERVISORY CONTROL WITH INFINITE-STRING SPECIFICATIONS 327
J.G. Thistle and W.M. Wonham

DECIDABILITY FOR TEMPORAL LOGIC IN CONTROL THEORY 335
J.F. Knight and K.M. Passino

II-B: LAYOUT AND WIRING

Chairman: M. Sarrafzadeh
Northwestern University

A LINEAR TIME ALGORITHM FOR CIRCULAR PERMUTATION LAYOUT 345
C.S. Rim, N.J. Naclerio, S. Masuda, and K. Nakajima

LAYOUTS OF GRAPHS OF ARBITRARY DEGREE 355
D.D. Sherlekar and J. JaJa

A LP APPROACH FOR OPTIMAL RECTANGLE ROUTING 365
Y.Y. Sung

SINGLE JOG RIVER ROUTING 373
T-C. Tuan and S.L. Hakimi

CHANNEL ROUTING WITH SUPERTERMINALS 375
G. Gudmundsson and S. Ntafos

A SUBOPTIMAL SOLUTION TO THE VIA PLACEMENT PROBLEM 377
T.F. Gonzalez and S. Kurki-Gowdara

THREE-LAYER WIRABILITY OF PLANAR LAYOUTS 387
T.F. Gonzalez and S-Q. Zheng

ON FINDING OBSTRUCTION SETS AND POLYNOMIAL-TIME ALGORITHMS FOR GATE MATRIX LAYOUT 397
R.L. Bryant, M.R. Fellows, N.G. Kinnersley, and M.A. Langston

II-C: ESTIMATION AND FILTERING

Chairman: J.B. Thomas
Princeton University

OPTIMAL JOINT PARAMETER AND STATE ESTIMATION OF LINEAR STOCHASTIC MIMO SYSTEMS USING PSEUDO-LINEAR IDENTIFICATION (PLID) 399
M.A. Hopkins and H.F. VanLandingham

A HYBRID METHOD OF ESTIMATING THE PARAMETERS OF ARMA MODELS 409
S. Li, Y. Zhu, and B.W. Dickinson

RELATIVE SUFFICIENCY 417
R.F. Popoli and J.M. Mendel

OPTIMAL AND SUBOPTIMAL FILTERING FOR LINEAR SYSTEMS DRIVEN BY SELF-EXCITED POISSON PROCESSES 426
M.A. Ingram and A.H. Haddad

AN ESTIMATION PROBLEM WITH A RANDOMLY PERTURBED LINEAR OPERATOR 436
M. Mobed and W.L. Root

SOME COMMENTS ON ESTIMATION OF RANDOM VARIABLES 446
A.E. Wessel and G.L. Wise

AN ERROR ANALYSIS FOR THE KALMAN FILTER APPLIED TO DATA CORRUPTED BY MULTIPLICATIVE NOISE 453
B.S. Chow and W.P. Birkemeier

II-D: IMAGE PROCESSING I: ANALYSIS AND SYNTHESIS

Chairman: D.C. Munson, Jr.
University of Illinois at Urbana-Champaign

EFFICIENT RECOGNITION OF HANDPRINTED CHARACTERS FROM TABLET-BASED INPUT 463
D.D. Kerrick and A.C. Bovik

USE OF FRACTAL PARAMETERS TO DETERMINE LOCAL SPATIO-TEMPORAL CHARACTERISTICS OF A TURBULENT POLLUTANT FROM VIDEO IMAGES 473
A. Ait-Kheddache and S.A. Rajala

DETERMINATION OF VELOCITY FIELDS FROM TIME-SEQUENTIALLY SAMPLED DATA 483
M. Rahgozar and J.P. Allebach

FAST ITERATIVE IMAGE RESTORATION ALGORITHMS 493
A.K. Katsaggelos and S.N. Efstratiadis

RESTORATION OF IMAGES DISTORTED BY SAMPLING JITTER 503
P. Shi and R.K. Ward

VECTOR DITHERING: BINARIZATION THROUGH UNCONVENTIONAL IMAGE PLANE TESSELLATION 513
T.S. Rao and G.R. Arce

II-E: LINEAR SYSTEMS

Chairman: P. Albertos
Univ. Politecnica de Valencia, Spain

LINEAR STABLE UNITY-FEEDBACK SYSTEM: NECESSARY AND SUFFICIENT CONDITIONS FOR STABILITY UNDER NONLINEAR ADDITIVE PLANT PERTURBATIONS 522
C.A. Desoer and M.G. Kabuli

EXTENDED ZEROS AND MODEL MATCHING 524
M.K. Sain and B.F. Wyman

TRANSFER MATRICES, POLYNOMIAL FRACTIONS AND REALIZATION OF CONTINUOUS-TIME LINEAR TIME-VARYING SYSTEMS 534
E. Emre, H-M. Tai, and J.H. Seo

CONDITIONS FOR EXISTENCE OF AN OPTIMAL CONTROL IN THE LQ REGULATOR PROBLEM FOR SINGULAR SYSTEMS 544
J.D. Cobb

STABILITY PARAMETERIZATIONS AND HIDDEN MODES IN TWO DEGREES OF FREEDOM CONTROL DESIGN 546
P.J. Antsaklis and O.R. Gonzalez

ON THE DISTANCE FROM AN UNCONTROLLABLE PAIR: A SUMMARY 556
M.A. Wicks and R.A. DeCarlo

MINIMUM EFFORT STATE FEEDBACK POLE-PLACEMENT FOR LINEAR MULTIVARIABLE SYSTEMS 558
N.F. Al-Muthairi and S.P. Bingulac

MEASURES OF CONTROLLABILITY AND OBSERVABILITY AND RESIDUES 560
D.K. Lindner, J. Babendreier, and A.M.A. Hamdan

OBSERVER POLE ASSIGNMENT USING EIGENVALUE SENSITIVITY 562
M.A. Choudhry and C.S. Sims

III-A: COMMUNICATIONS AND CODING

Chairman: W.E. Stark
University of Michigan

WORST-CASE ANALYSIS OF SPREAD-SPECTRUM SYSTEMS WITH OUTPUT QUANTIZATION 564
M.V. Hedge, W.E. Stark, and D. Teneketzis

DYNAMIC CLIPPING FOR DETECTION OF FH/BFSK SIGNALS WITH DIVERSITY 574
Y.K. Kim and C.C. Lee

AN EXPURGATED LOWER BOUND ON THE FREE EUCLIDEAN DISTANCE OF TRELLIS CODES 584
M. Rouanne and D.J. Costello, Jr.

RANDOM CODING BOUNDS FOR A FAMILY OF BALANCED ERROR-CORRECTING CODES 594
M.A. Herro and V. Telang

ON GENERALIZED ERROR CORRECTING TRELLIS CODES WITH BALANCED BINARY SYMBOLS 596
H.C. Ferreira, D.A. Wright, and A.L. Nel

EFFICIENT MAXIMUM-LIKELIHOOD DECODING OF THE EXTENDED NORDSTROM-ROBINSON CODE 598
A.D. Abbaszadeh and C.K. Rushforth

SOFT DECISION DECODING OF CONCATENATED CODES 600
A.A. Hassan and W.E. Stark

EFFICIENT ALGORITHMS FOR THE SEARCH OF ZERO-CONCURRING CODEWORDS 610
A. Vardy and Y. Be'ery

CYCLIC CODES AND THEIR WEIGHTS 620
V. Pless

PERFORMANCE OF HYBRID ARQ CODING FORMATS OVER A BINARY SYMMETRIC CHANNEL 622
G.M. Huffman and F.D. Garber

III-B: PARALLEL AND DISTRIBUTED COMPUTATION

Chairman: G.N. Frederickson
Purdue University

A SIMPLE PARALLEL TREE CONTRACTION ALGORITHM 624
K. Abrahamson, N. Dadoun, D.G. Kirkpatrick, and T. Przytycka

AN EFFICIENT ASYNCHRONOUS DISTRIBUTED PROTOCOL TO TEST FEASIBILITY OF THE DUAL TRANSSHIPMENT PROBLEM 634
M.A. Comeau, K. Thulasiraman, and K.B. Lakshmanan

UNIVERSAL RANKINGS FOR BROADCASTING IN TREE NETWORKS 641
A. Rosenthal and P. Scheuermann

MINIMIZING SCHEDULE LENGTH SUBJECT TO MINIMUM FLOW TIME 650
J.Y-T. Leung and G.H. Young

AN OPTIMAL PARALLEL SOLUTION TO THE ECDF SEARCHING PROBLEM FOR HIGHER DIMENSIONS ON A MESH-OF-PROCESSORS 660
F. Dehne and I. Stojmenovic

SPORT-SORT SORTING ALGORITHMS AND SPORT TOURNAMENTS 662
H. Block

III-C: STOCHASTIC SYSTEMS

Chairman: R. Mazumdar
Columbia University

MODELING AND DYNAMIC SCHEDULING OF A QUEUEING SYSTEM WITH BLOCKING AND STARVATION 664
R.A. Milito and H. Levy

STRUCTURAL DECOMPOSITION OF MULTIPLE TIME SCALE MARKOV PROCESSES 674
J.R. Rohlicek and A.S. Willsky

THE SETTLING TIME OF A CLOSED JACKSON NETWORK 684
V. Anantharam

ON SCHEDULING TASKS WITH STOCHASTIC PROCESSING TIMES ON PARALLEL SERVERS 694
S.P.R. Kumar

GLOBAL CONVERGENCE OF RECURSIVE STOCHASTIC MINIMIZATION ALGORITHMS 700
S. Gelfand and S.K. Mitter

A GLOBAL OPTIMAL CONTROL SETTING FOR LARGE DEVIATIONS 707
E. Kappos

OPTIMAL CONTROL OF ITEMIZED PRODUCTION TO MEET UNCERTAIN DEMAND 713
J.B. Ribeiro do Val

STOCHASTIC APPROXIMATION FOR DISTRIBUTED PARAMETER SYSTEMS 721
N. Berman and A. Shwartz

III-D: IMAGE PROCESSING II: EDGE DETECTION AND RANKED-ORDER FILTERING

Chairman: T.S. Huang
University of Illinois at Urbana-Champaign

EDGE DETECTION BY COST MINIMIZATION 731
H.L. Tan, E.J. Delp, and S.B. Gelfand

REGULARIZED DIFFERENTIATION FOR EDGE DETECTION 741
A.K. Katsaggelos and B.J. Sullivan

EDGE ENHANCEMENT USING ORDER STATISTIC LAPLACIAN 743
Y.H. Lee and S.Y. Park

MULTILEVEL MEDIAN FILTERS 753
G.R. Arce and R.E. Foster

A FAST RECURSIVE TWO-DIMENSIONAL MEDIAN FILTERING ALGORITHM FOR MULTI-LEVEL IMAGE ROOT EXTRACTION 763
W.W. Boles, M. Kanefsky, and M. Simaan

CENSORED ORDER STATISTIC FILTERS 773
L. Naaman and A.C. Bovik

AN ANALYSIS OF A MULTIPLE MODEL MORPHOLOGICAL FILTER 775
J. Song and E.J. Delp

III-E: ADAPTIVE SYSTEMS

Chairman: J. Dorsey
Georgia Institute of Technology

DECENTRALIZED ADAPTIVE CONTROL OF LARGE SCALE SYSTEMS 785
H. Yousef and M. Simaan

ADAPTIVE CONTROL UNDER PARTIALLY MEASURED CONTROL VARIABLES 795
P. Albertos and F. Morant

DISTURBANCE-GENERATED BIFURCATION IN A PROTOTYPE ADAPTIVE SYSTEM WITH e_1-MODIFICATION LAW 801
S. Bai and F.M.A. Salam

ADAPTIVE CONTROL OF A CLASS OF DIFFERENTIAL DELAY SYSTEMS 811
F. Pourboghrat

SLIDING MODE PROPERTIES OF MIN-MAX CONTROLLERS AND OBSERVERS FOR UNCERTAIN SYSTEMS 813
S.H. Zak and R.A. DeCarlo

IV-A: ROBOTICS

Chairman: M.W. Spong
University of Illinois at Urbana-Champaign

ON THE STABILITY OF A MANIPULATOR INTERACTING WITH ITS ENVIRONMENT 821
J.E. Colgate and N. Hogan

CLOSED CHAIN MANIPULATOR DYNAMICS AND CONTROL 829
J.J. Murray and G.H. Lovell

PLANNING A COLLISION - AVOIDANCE PATH USING ANALYTIC MODELING FOR OBSTACLES 839
Y.M. Enab and J.Y.S. Luh

STABILIZATION OF UNCERTAIN SYSTEMS SUBJECT TO HARD BOUNDS ON CONTROL WITH APPLICATION TO A ROBOT MANIPULATOR 841
M. Hached, S.M. Madani-Esfahani, and S.H. Zak

FINDING EFFECTIVE 'FORCE TARGETS' FOR TWO-DIMENSIONAL, MULTIFINGER FRICTIONAL GRIPS 843
J.T. Schwartz and M. Sharir

IV-B: COMPUTATIONAL GEOMETRY

Chairman: H. Edelsbrunner
University of Illinois at Urbana-Champaign

A POLYGONAL APPROACH TO HIDDEN-LINE ELIMINATION 849
M.T. Goodrich

A DUAL APPROACH TO DETECT POLYHEDRAL INTERSECTIONS IN ARBITRARY DIMENSIONS 859
O. Gunther and E. Wong

FINDING LARGEST INSCRIBED EQUILATERAL TRIANGLES AND SQUARES 869
N.A. De Pano, Y. Ke, and J. O'Rourke

A NEW APPROACH TO THE DYNAMIC MAINTENANCE OF MAXIMAL POINTS IN A PLANE 879
G. Frederickson and S. Rodger

THE RECTILINEAR VORONOI DIAGRAM WITH BARRIERS 889
D.R. Fowler and J.M. Keil

SOME RESTRICTED RECTANGLE COVERING PROBLEMS 898
H.E. Conn and J. O'Rourke

AN EFFICIENT HEURISTIC ALGORITHM FOR A 3-SPACE n-PARTITIONING PROBLEM USED IN POLYGON RETRIEVAL AND NON-PARAMETRIC STATISTICS 908
P. Gburzynski and M. McLeish

IV-C: INFORMATION THEORY AND SOURCE CODING

Chairman: R.E. Blahut
IBM Corp. and University of Illinois at Urbana-Champaign

A COMPARISON OF CONDITIONS FOR POSITIVE CAPACITY OF AN ARBITRARILY VARYING CHANNEL 917
I. Csiszar and P. Narayan

MULTITERMINAL SOURCE ENCODING WITH ONE DISTORTION CRITERION 919
T. Berger and R. Yeung

HIGH-ORDER MARKOV ARITHMETIC CODING FOR DATA COMPRESSION 928
Y. Bar-Ness and S. Choi

ROBUST ADAPTIVE BUFFER-INSTRUMENTED ENTROPY-CODED QUANTIZATION OF STATIONARY SOURCES 938
J.W. Modestino, D.D. Harrison, and N. Farvardin

A PAIR OF INFORMATION THEORETIC LEMMAS WITH APPLICATION TO RUN-LENGTH CODING 946
C. Heegard

A COMPARISON OF THE WOLF-ZIV AND ALPHABET-CONSTRAINED APPROACHES FOR VECTOR QUANTIZATION OF NOISY SPEECH 955
J.D. Gibson, T.R. Fischer, and B. Koo

RATE-DISTORTION PERFORMANCE OF 2-D DPCM FOR GAUSSIAN AUTOREGRESSIVE SOURCES 965
J.W. Modestino and F. Azadegan

HIGHLY EFFICIENT ARITHMETIC CODING SCHEMES FOR LINE DRAWINGS BASED ON CHAIN CODE REPRESENTATIONS 967
J.G. Dunham and C-C. Lu

IV-D: COMMUNICATION NETWORKS

Chairman: K.S. Vastola
Rensselaer Polytechnic Institute

DECOMPOSITION OF A MULTIPLE ACCESS NETWORK INTO A NUMBER OF INTERCONNECTED SUBNETWORKS 969
M.A. Karatzoglu and A. Ephremides

DECENTRALIZED ALGORITHMS FOR OPTIMAL FLOW CONTROL 979
A.D. Bovopoulos and A.A. Lazar

ON PARETO OPTIMAL FLOW CONTROL IN A MULTICLASS ENVIRONMENT 989
C. Douligeris and R. Mazumdar

A (NOT VERY) SIMPLE ROUTING PROBLEM 998
P.P. Bhattacharya, I. Viniotis, and A. Ephremides

TERMINATION DETECTION IN A BROADCAST NETWORK 1007
T.H. Lai and T.H. Yang

ON THE PERFORMANCE OF SW-BANYAN NETWORKS 1017
M.H. Peyravi and M.R. Meybodi

THE COMPLEXITY OF ROUTING THROUGH AN OMEGA NETWORK 1027
P.J. Bernhard and D.J. Rosenkrantz

TWO-STAGE REALIZATIONS OF SYMMETRIC GROUPS 1037
A.Y. Oruc and S. Schneider

IV-E: TESTING AND FAULT TOLERANCE

Chairman: J.H. Patel
University of Illinois at Urbana-Champaign

TEST GENERATION FOR C-TESTABLE ARRAY MULTIPLIERS 1040
S-M. Chang and C-L. Wey

A FUNCTIONAL TEST SET FOR CMOS CIRCUITS 1050
G. Gupta and N.K. Jha

ON THE DESIGN OF PARALLEL SCAN TESTABLE FINITE STATE SEQUENTIAL MACHINES 1059
S. Iyengar, R. Dandapani, and S.M. Reddy

TRANSITION COUNT AS A COMPRESSION FUNCTION IN BUILT-IN SELF-TEST ENVIRONMENT 1069
K. Akiyama and K.K. Saluja

V-A: RANDOM ACCESS COMMUNICATIONS

Chairman: S. Verdu
Princeton University

MAXIMUM FINDING IN MULTIPLE ACCESS NETWORKS 1079
C.U. Martel and T.P. Vayda

COLLISION RESOLUTION ALGORITHMS FOR A TIME-CONSTRAINED MULTIACCESS CHANNEL 1081
S.S. Panwar, D. Towsley, J.K. Wolf, and Y. Armoni

ON DECENTRALIZED CONTROL ALGORITHMS FOR MULTIPACKET ALOHA 1089
S. Ghez, S. Verdu, and S.C. Schwartz

STABILITY CONDITIONS IN A MULTIQUEUE SYSTEM WITH APPLICATIONS TO THE ANALYSIS OF COMPUTER COMMUNICATION NETWORKS 1099
W. Szpankowski and V. Rego

THROUGHPUT ANALYSIS OF MAXEMCHUK'S INTEGRATED VOICE/DATA LOCAL AREA NETWORK PROTOCOL 1101
R. Kohli, A.C. Mingo, and K.S. Vastola

THE EFFECT OF EVENLY DISTRIBUTED STATIONS ON CARRIER SENSE MULTIPLE ACCESS PROTOCOL 1103
J.M. Vanderwiele and J.G. Bredeson

V-B: NEURAL NETWORKS

Chairman: V. Anantharam
Cornell University

QUALITATIVE ANALYSIS OF NEURAL NETWORKS: GLOBAL THEORY 1113
J-H. Li, A.N. Michel, and W. Porod

QUALITATIVE ANALYSIS OF NEURAL NETWORKS 1123
A.N. Michel, J.A. Farrell, and W. Porod

CONVERGENCE ISSUES IN PERCEPTRON BASED ADAPTIVE NEURAL NETWORK MODELS 1133
Y. Shrivastava and S. Dasgupta

ON THE DESIGN OF NEURAL NETS 1142
F.M.A. Salam, M.R. Choi, Y. Wang, and B. Javidi

V-C: STABILITY THEORY AND OPTIMAL CONTROL

Chairman: D.K. Lindner
Virginia Polytechnic Institute

A NEW INCLUSION CONCEPT AND OPTIMAL CONTROL FOR SYSTEMS WITH OVERLAPPING INFORMATION SETS 1152
A. Iftar and U. Ozguner

DISTRIBUTED COMPUTATIONS OF OPTIMAL CONTROL STRATEGIES FOR A CLASS OF INTERCONNECTED SYSTEMS 1162
L. Acar and U. Ozguner

A STABILITY THEORY OF LINEAR TIME VARYING SYSTEMS 1172
P.R. Bouthellier and B.K. Ghosh

FEEDBACK SYSTEMS WITH TAPERED AND OTHER SPECIAL INPUT SPACES 1181
W.L. Root and D. Serakos

PARALLEL GRADIENT PROJECTION ALGORITHMS TO SOLVE THE DISCRETE LQR OPTIMAL CONTROL PROBLEM WITH HARD CONTROL BOUNDS 1190
G.G.L. Meyer and L.J. Podrazik

CONJUGATE POINTS IN OPTIMAL CONTROL 1200
J. Kogan

TIME SENSITIVITY ANALYSIS OF NONCONVENTIONAL SAMPLED-DATA SYSTEMS 1202
P. Albertos and J. Tornero

V-D: FOUNDATIONS OF COMPUTING

Chairman: D.E. Muller
University of Illinois at Urbana-Champaign

INTRODUCTION TO DIALECTICAL NETS 1204
R.E. Kent

ON EXACT, CRITICAL AND UNIQUE VERSIONS OF SOME GRAPH PROBLEMS 1214
L.E. Rosier and H-C. Yen

COMPLEXITY AND DECIDABILITY FOR RESTRICTED CLASSES OF PICTURE LANGUAGES 1224
C. Kim

STRUCTURAL PROPERTIES OF COMMUNICATION STRUCTURES GENERATED BY AGGREGATE REWRITING GRAPH GRAMMARS 1234
D.A. Bailey and J.E. Cuny

INDEX OF AUTHORS

A Simple Parallel Tree Contraction Algorithm (Preliminary Version)[†]

K. Abrahamson,
Computer Science Dept., Washington State University, Pullman, Washington 99164, USA

N. Dadoun, D.G. Kirkpatrick, T. Przytycka
Computer Science Dept., University of British Columbia, Vancouver, BC, Canada V6T 1W5

Abstract

A simple reduction from the tree contraction problem to the list ranking problem is presented. The reduction takes O(log n) time for a tree with n nodes, using O(n/log n) EREW processors. Thus tree contraction can be done as efficiently as list ranking.

A broad class of parallel tree computations to which the tree contraction techniques apply is described. This subsumes a number of earlier characterizations.

1. Introduction

The tree contraction problem is to reduce a rooted ordered tree[1] to its root by a sequence of independent vertex removals. Tree contraction can be viewed as a general technique for scheduling parallel computations on trees. We describe an efficient parallel algorithm for tree contraction based on a simple reduction to list ranking. The reduction for trees with n nodes can be implemented on an exclusive read - exclusive write (EREW) PRAM (see [V83] for a detailed comparison of parallel models of computation) in O(log n) time and using O(n/log n) processors. Combined with recent results on parallel list ranking, it leads to a simple optimal parallel tree contraction algorithm.

Parallel algorithms for tree contraction have been studied before in several papers. Most of the earlier results have been described for the stronger concurrent read/exclusive write (CREW) model though they do not appear to make any essential use of concurrent reads. Miller and Reif [MR85] describe a deterministic algorithm which runs in O(log n) time with O(n) processors. They present their algorithm in the context of dynamic expression evaluation for an expression presented in the form of a parse tree. A single step of their algorithm converts a current binary parse tree to a simpler one by removing in parallel all leaves (RAKE operation) and compressing maximal chains of nodes with only one child (COMPRESS operation). They show that after O(log n) such steps a given tree is reduced to its root.

Miller and Reif apply their method to construct parallel algorithms for problems which can be reduced to computation on trees. They give a randomized algorithm for testing isomorphism of trees, a deterministic algorithm for constructing a tree of 3-connected components of a planar graph, and other algorithms.

He [H86] defines the binary tree algebraic computation (BTAC) problem and applies Miller and Reif's technique to obtain a parallel algorithm for this problem. Roughly speaking the BTAC problem is to compute the value of an algebraic expression given in the form of a parse tree under the assumption that the algebra in which the computation is performed has a finite carrier.

[†] This research was supported in part by the Natural Sciences and Engineering Research Council of Canada.

[1] Throughout this paper trees are all rooted and ordered. These adjectives will not be repeated hereafter.

Another approach to dynamic expression evaluation is presented by Gibbons and Rytter [GR86]. Their algorithm runs in O(log *n*) time using O(*n*/log *n*) processors. They assume, however, that the input (the string representing the expression to be computed) is given in an array and is hence preordered. Similarly to He, they prove that the algorithm can be applied to compute an algebraic expression in any algebra with finite carrier.

Cole and Vishkin [CV86c] propose an alternative method for computation on trees. They solve the tree contraction problem by parallel reduction to the list ranking problem. In this respect their approach is similar to ours. Our reduction, an abstraction of the technique used for the parallel preprocessing of region trees for fast (sequential) subdivision search [DK87], is simpler and more explicit than that of Cole and Vishkin; in particular, it completely avoids the centroid decomposition techniques that lie at the heart of their reduction.

2. Reduction to list ranking

Within a linked list, the *rank* of a given element is defined as the number of elements following that element. Given a linked list, the *list ranking problem* is to determine in parallel the rank of each element in the list. Once each element has its rank, an ordered array of elements can be constructed simply by using the rank as an array index. The list ranking problem generalizes easily to the weighted list ranking problem, where each element has an associated weight and the goal is to determine the *weighted rank* — the sum of the weights of all subsequent elements — for each list element.

The list ranking problem has received much attention in recent research (*cf.* [ADKP87a], [CV86a], [CV86b]) because of its fundamental role in many parallel algorithms. Of particular interest is its role in the so-called *Euler Tour Technique* [TV84] which can be used to compute various functions on trees including preorder number, postorder number, descendent numbering and others.

Among the important contributions of Miller and Reif [MR85] is their abstraction of the problem of tree contraction. This leads to a separation of the problem from its familiar applications (notably, dynamic expression evaluation) and places it, along with list ranking, among the fundamental problems of parallel computation. A natural side effect has been the identification of new and unforeseen applications [DK87, H86, GR86]. To further solidify this abstraction we present the following definitions.

We will assume that trees are presented as an (unordered) array of vertices each of which has associated with it a parent pointer and a doubly-linked list of children. (Of course, it is possible to efficiently convert to or emulate such a representation starting with other more primitive representations.) (Successive) vertices on any list of children are said to be (immediate) siblings.

Let T be any binary tree with vertex set $V(T)$. A sequence of trees $T_1, T_2, \dots T_k$ is said to be a *tree contraction sequence* of length k for T if,

(i) $T_1 = T$;

(ii) $V(T_i) \subseteq V(T_{i-1})$;

(iii) $|V(T_k)| \leq 3$; and

(iv) if $v \in V(T_{i-1}) - V(T_i)$ then either

(a) v is a leaf of T_{i-1}, or

(b) v has exactly one child, x, in T_{i-1}, $x \in V(T_i)$, and the parent of v in T_{i-1} is the parent of x in T_i.

It is clear from the definition that successive trees in a tree contraction sequence are formed by "independent" executions of the following two fundamental operations (see Figure 2.1): Prune(v) — leaf v is removed from the current tree; and Bypass(v) — non-root node v with exactly one child x is removed from the current tree, and the parent w of v becomes the new parent of x (with x replacing v as a child of w).

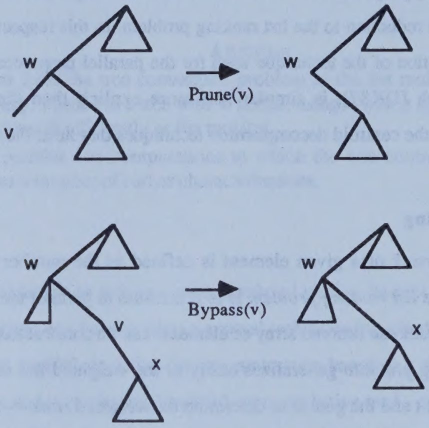


Figure 2.1. Prune and Bypass operations

By "independent" we mean that if v is pruned or bypassed then its parent is not bypassed. In this way tree modifications are ensured to be local and executable in parallel.

It is clear that every binary tree with n nodes admits a contraction sequence of length n; simply prune $n - 1$ times in succession until the tree has been reduced to its root. On the other hand, since just over half of the nodes can be removed in any one step, every contraction sequence must have length at least $\log(n) - 1$. We say that a contraction sequence is *optimal* if it has length $O(\log n)$. (Note, it may not be immediately obvious that every binary tree has an optimal contraction sequence, let alone that such a sequence can be constructed efficiently.)

The *tree contraction problem* is to construct, for an arbitrary binary tree T, an optimal tree contraction sequence for T. It is not necessary to construct the sequence explicitly; it suffices to associate with each node v the index i of the highest indexed tree containing v in the sequence, together with pointers to the parent and child (if any) of v in T_i. It is meaningful to talk about tree contraction for non-binary trees as well. While a number of possible definitions suggest themselves the most natural is probably that which arises in conjunction with the familiar interpretation of general trees as regular binary trees (*cf.* Knuth[K68], pp. 332-345). Suppose T is an arbitrary ordered rooted tree. Consider the tree T' constructed from T as follows: If v is a vertex of T with d children then the vertex set of T' includes $v^1, v^2, \ldots, v^{d+1}$. Vertex v^{i+1} is the right child of vertex v^i in T', for $1 \leq i \leq d$. Furthermore, if vertex w is the i^{th} child of v in T then vertex w^1 is the left child of vertex v^i in T' (see Figure 2.2).

Note that T' is a regular binary tree (all of its internal nodes have exactly two children). Furthermore, there is no cost associated with the construction of T' from T since it involves only a reinterpretation of pointers. Any tree contraction sequence for T' yields — what we might define to be — a tree contraction sequence for T.

A vertex v of T is pruned or bypassed at step i if the last of its associated vertices $v^1, \ldots, v^{d+1}$ in T' is pruned or bypassed at step i. A straightforward case analysis shows that the resulting sequence satisfies the additional "independence" condition that no two adjacent siblings are removed simultaneously. Since T' has fewer than twice as many vertices as T, it follows that any optimal tree contraction sequence for T' yields an optimal tree contraction sequence for T. (This reduction of general tree contraction to binary tree contraction is similar to the "binarization" step of Cole and Vishkin [CV86c].)

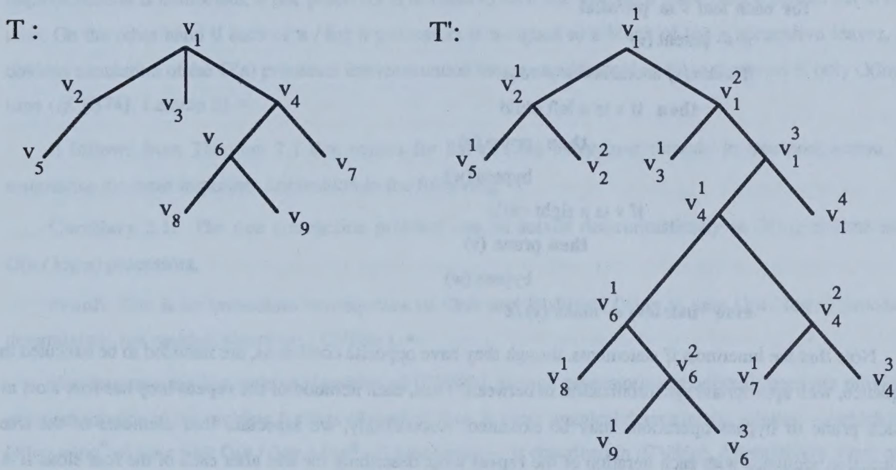


Figure 2.2 Interpretation of Trees as Binary Trees

Suppose once again that T is a binary tree. It is clear that if T has height h then T admits a contraction sequence of length h consisting entirely of prunes but when T is unbalanced such a sequence is far from optimal. Miller and Reif [MR85] introduce the idea of path compression — a familiar operation in the context of list ranking — to cope with trees, or parts thereof, that have become so unbalanced that they have degenerated into paths. Miller and Reif show that by interleaving global pruning (that is, the removal of *all* leaves) with global path compression (the compression of *all* paths of length greater than one) it is possible to construct an O(log n) contraction sequence in O(log n) time using O(n) processors. The complications of Miller and Reif's approach, especially with regard to processor reduction, arise in the compression step. Our approach differs in that we define the primitive contraction operations at a more elementary level. By scheduling the order in which leaves are eliminated it is possible to completely eliminate the difficulties associated with path compression. Indeed, paths of length greater than two never arise as our algorithm contracts an (initially regular) binary tree.

The pair of operations prune(v) followed by bypass(parent(v)) (where v is any leaf) form a conceptual unit in our algorithm. The algorithm proceeds in phases, each of which consists of a batch of these basic contractions performed in parallel. The independence of the underlying operations is guaranteed by a simple global schedule for leaf removal. Let the leaves be numbered in left to right order. A leaf is removed in phase t if the rightmost 1 in its leaf index is in position t.

Our binary tree contraction algorithm has the following simple description:

```
procedure contract ($T$)
(* Assign leaf indices from 0 to $n$ - 1 *)
for each leaf $v$ in parallel
    index($v$) ← left_to_right leaf index of $v$
(* Contraction iterations. *)
repeat $\lceil \log n \rceil - 1$ times
    for each leaf $v$ in parallel
        $w$ ← parent ($v$)
        if index($v$) is odd and $w \neq$ root
            then if $v$ is a left child
                    then prune ($v$)
                         bypass ($w$)
                 if $v$ is a right child
                    then prune ($v$)
                         bypass ($w$)
        else index($v$) ← index ($v$) /2
```

Note that the innermost **if** statements, though they have opposite conditions, are intended to be executed in sequence, with appropriate synchronization in between. Thus, each iteration of the **repeat** loop has four slots in which prune or bypass operations may be executed. Accordingly, we associate four elements of the tree contraction sequence with each iteration of the **repeat** loop, describing the tree after each of the four slots. It is also helpful to view the behaviour of the algorithm at two other levels. It is immediate from the description that each prune operation is immediately followed by a bypass. Hence in each successive pair of slots a number of composite prune-bypass operations are executed in parallel. Each pair of these composite slots (making up an entire iteration of the **repeat** loop) serves to eliminate all of the leaves with odd parity in the current tree together with their parents.

Lemma 2.1. Procedure **contract** constructs an optimal tree contraction sequence.

Proof: It suffices to demonstrate that the prunes and bypasses are performed independently. Since prunes and bypasses are never executed simultaneously, it need only be demonstrated that no vertex v and its parent w are ever bypassed simultaneously. Suppose this is not the case. Without loss of generality, v is the right child of w. Since they are bypassed simultaneously, they must both have leaves as left children. But since these leaves are adjacent in the left to right order and since the index array maintains each leaf's left-to-right rank in the current contracted tree, v and w must have indices of opposite parity, a contradiction. •

Theorem 2.1. Procedure **contract** provides an O(log n) time and O(n / log n) processor deterministic reduction of tree contraction to list ranking.

Proof: First note that there is a straightforward reduction of the leaf ranking problem, which arises in the first step of procedure **contract**, to the list ranking problem. The method is called the Euler Tour technique [TV84]. Each node v of T is split into three nodes v_T, v_L and v_R. For each of the resulting nodes we define a *next* field as follows. If v is a leaf then $v_T.next = v_L$ and $v_L.next = v_R$. If w is the right child of v then

v_L.next = w_T and w_R.next = v_R. If w is the left child of v then v_T.next = w_T and w_R.next = v_L. What results is a list that starts at root$_T$ and ends at root$_R$ and traverses each edge of T once in each direction. If we assign weight(z) = 1 if $z = v_T$ and v is a leaf and weight(z) = 0 otherwise, then the weighted rank of each node v_T, where v is a leaf, gives the leaf index of v in T.

As a consequence of the above, it suffices to prove that the iterated contraction step of procedure **contract** can be implemented in O(log n) time using O(n/log n) processors. An O(log n) time, O(n) processor implementation is immediate; if one processor is devoted to each leaf then each phase can be carried out in O(1) time. On the other hand if each of n / log n processors is assigned to a block of log n successive leaves, the obvious simulation of the O(n) processor implementation incurs an additional additive overhead of only O(log n) time (*cf.* [B74], Lemma 2). •

It follows from Theorem 2.1 that results for list ranking carry over directly to tree contraction. We summarize the most important implication in the following:

Corollary 2.1. The tree contraction problem can be solved deterministically in O(log n) time using O(n / log n) processors.

Proof: This is an immediate consequence of Cole and Vishkin's O(log n) time O(n / log n) processor deterministic list ranking algorithm [CV86b]. •

The deterministic list ranking algorithm of [CV86b], though asymptotically optimal, does not provide a practical solution to the problem for lists of realistic size. A more practical deterministic solution — which uses O(log n log* n) time with O(n / (log n log* n)) processors — is described in [CV86a]. Alternatively, there exist practical randomized parallel list ranking algorithms that achieve the asymptotically optimal bounds [MR85, ADKP87a].

3. Algebraic tree computations

The applications presented in this section serve to illustrate our tree contraction algorithm. In particular, we will use the explicit sequence of trees created by the algorithm in section 2 and treat each prune and its subsequent bypass as an indivisible operation. Though the descriptions benefit from the simplicity of our particular approach to tree contraction, it is clear that with suitable modifications other tree contraction schemes may be used for these applications.

A tree contraction algorithm gives a method for solving a large class of parallel tree computation problems. This class includes, for example, dynamic expression evaluation ([MR85], [GR86]). He [H86] and Gibbons and Rytter [GR86] noted that any algebraic expression with operands from an algebra with carrier of fixed finite size can be computed in the cost of tree contraction. This result provides efficient parallel algorithms for several optimization problems, for example minimum covering set, maximum independent set and maximum matching, when the underlying graph is a tree [H86].

In fact, we can relax the assumption that the carrier of the algebra is of fixed finite size and put some restrictions on the operations only. We can generalize He's binary tree algebraic computation problem in the following way: Let S be a set and $F \subseteq \{f \mid f: S \times S \rightarrow S\}$ a set of two-variable functions over S. The objective of bottom-up algebraic tree computations is to take any regular binary tree T whose leaves are labelled by elements of S and whose internal nodes are labelled by elements of F and to evaluate the algebraic expression associated with T (where functions at internal nodes denote operators and elements labelling leaves are operands).

It is natural (and helpful) to generalize the above notion to include a set of functions $G \subseteq \{g \mid g : S \to S\}$, including the identity function, which serve as edge labels and influence the computation in the obvious way. The triple (S, F, G) defines a *bottom-up algebraic tree computation* (B-ATC) *problem.* Such an problem is said to be *decomposable* if

(i) the sets F and G are indexed sets and their elements can be evaluated in O(1) sequential time; and

(ii) for all $g_i, g_j, \in G, f_m \in F$ and $a \in S$, the functions g_s and g_t given by $g_s(x) = g_i(f_m(g_j(x), a))$ and $g_t(x) = g_i(f_m(a, g_j(x)))$ both belong to G and their indices s and t can be computed in O(1) sequential time from i, j, m and a.

Condition (ii) defines a kind of closure operation on the sets F and G. Its significance is perhaps most easily understood by referring to the transformations of Figure 3.1.

Note that any algebra with a finite carrier has associated with it a decomposable B-ATC problem. It is also easy to see that general {+, -, *, /} arithmetic computations can be described as a decomposable B-ATC problem [B74], provided arithmetic operations are assumed to take constant time.

Theorem 3.1: If (S, F, G) is a decomposable B-ATC problem then for any input tree the associated algebraic expression can be evaluated in time proportional to the cost of tree contraction using the same number of processors.

Proof: Condition (ii) of decomposability guarantees that each of the trees in the tree contraction sequence of T can be labelled in such a way that their associated algebraic expression is equivalent to that of T. Since this labelling can be computed in O(1) time at each step the entire computation runs in time proportional to that of tree contraction.•

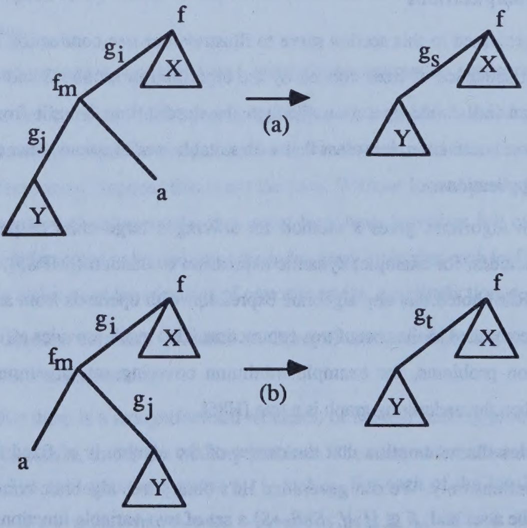


Figure 3.1

Remark 3.1: We should note that we can easily modify the general computation scheme in such a way that we will compute also all the functions in internal nodes. In order to do so each bypassed node should maintain a pointer to the lower level endpoint of the bypassing edge (i.e. the node whose computation has to be finished in order to finish computation in the given node). For example, in Figure 3.1 the bypassed node labelled f_m retains a pointer to the leaf labelled *a* and the root of the subtree *Y*. After finishing the computation of the function in the root we add a new phase to the algorithm. In this phase we allow all the internal nodes to finish computation of the values of the function assigned to these nodes. The computation is completed at vertices in the reverse of their order of elimination in the contraction sequence.

Remark 3.2: Let us assume that we have computed the value of the expression defined by a given tree *T*. Assume now that we would like to compute an expression defined by the same expression tree except that one leaf has a new value. To compute this modified expression we can use all the partial results from the previous computation with the exception of the values on the path from the leaf to the root. We can do it in O(log *n*) *sequential* time using information stored in the sequence of trees constructed by the contraction algorithm. We need to make only one change in each of the trees. By conditions (i) and (ii) we need constant time to correct each of them. There are O(log *n*) trees so the algorithm runs in O(log *n*) time. As an easy generalization, we note that in order to recompute the value of the expression when values in *l* leaves have been changed we need O(log *n*) time with O(min(*l*, *n*/log *n*)) processors.

The B-ATC problem provides a useful abstraction of many bottom-up tree-based computations. In a number of applications it is necessary to consider top-down (or perhaps a combination of bottom-up and top-down) computations based on trees. Let *S* be a set as before and let $H \subseteq \{h \mid h : S \to S\}$ be a set of one-variable functions over *S*. The objective of a top-down algebraic tree computation is to take any regular binary tree *T* whose root is labelled by an element of *S* and whose non-root nodes are labelled by elements of *H* and to evaluate the function associated with each node in the natural way (the root takes its given value and the value of a non-root node is its associated function applied to its parent's value).

As with the B-ATC problem, it is natural to generalize this notion by associating elements of *H* (now assumed to include the identity function) with the edges of *T* as well; the resulting computation are modified in the obvious way. A pair (*S*, *H*) defines a *top-down algebraic tree computation* (T-ATC) *problem*. Such a problem is said to be *decomposable* if

(i) the set *H* is indexed and its elements can be evaluated from their index and argument in O(1) sequential time; and

(ii) *H* is closed under composition and for each $h_i, h_j \in H$ the index of $h_i \circ h_j$ can be computed in O(1) sequential time from *i* and *j*.

Theorem 3.2 If (*S*, *H*) is a decomposable T-ATC problem then for any input tree *T* the value associated with each vertex can be computed in time proportional to the cost of tree contraction, using the same number of processors.

Proof: By the decomposability condition, each edge *e* introduced by a bypass operation can, as part of the tree contraction process, be labelled by the composition of the functions associated with the nodes and edges on the path (in *T*) joining the endpoints of *e*. The vertices of *T* are considered in the opposite order of their elimination in the tree contraction sequence. It is straightforward to confirm that the value of all vertices in tree T_i can be computed in O(1) time knowing the values of all vertices in tree T_{i-1}. •

4. Discussion

The optimal reduction of tree contraction to list ranking allows us to construct a simple tree contraction algorithm which runs in the cost of the chosen list ranking algorithm. As a consequence, we have an optimal solution to parallel tree contraction using the list ranking results of Cole and Vishkin [CV86b]

Gibbons and Rytter [GR86] have presented an optimal algorithm for dynamic evaluation of algebraic expressions assuming that the input is given in an array. They apply the algorithm of Bar-On and Vishkin [BV85] to obtain a parse tree. The leaves of the parse tree are numbered from left to right on the basis of their index in the input array. This numbering makes it possible to group leaves in blocks of O(log *n*) size and then to perform computation in each block in parallel. As a result the parse tree is contracted to a tree of size O(*n*/log *n*), at which point the algorithm of Reif and Miller computes the tree contraction using an optimal number of processors.

If we assume that the input is given in the form of a binary tree instead of an array the algorithm of Gibbons and Rytter can still be applied by numbering the leaves of the tree in a preprocessing step. As in our algorithm, this can be done in the cost of list ranking using the Euler tour technique [TV84]. An advantage of our approach is the clear identification of the role played by list ranking in tree contraction. In our approach the method in which a tree is contracted does not depend on the actual size of the tree. In this respect, our approach is more similar to that of Cole and Vishkin [CV86c].

Cole and Vishkin's tree contraction algorithm uses the same two basic parallel operations as ours. The first stage of Cole and Vishkin's algorithm is to compute for each tree vertex *v* the function SIZE(*v*), which is equal to the number of nodes in the subtree rooted at *v*. This step is done using the Euler tour technique. Then, using function SIZE, the tree is partitioned into the so-called centroid paths. The bypass operation can be applied only to a node which does not have a non centroid child. The next step of the algorithm (called the scheduling stage) is to compute the order in which the prune and bypass operations should be performed. In the last stage, called the evaluation stage, the prune and bypass operations are performed according to the order computed in the previous stage. In the evaluation stage some work has to be done in order to assign processors to the jobs.

In our algorithm the reduction to list ranking is done in a simpler way. The only information which is used to schedule the order of performing bypass operations is the initial numbering of the leaves. Consequently we do not need a separate scheduling phase, which in Cole and Vishkin's algorithm is quite complicated. Also the assignment of processors to the jobs is simpler in our approach.

In the full version of this paper [ADKP87b], we show how the tree contraction algorithm can be applied to solve efficiently several problems on a class of tree representable graphs: complement reducible graphs or cographs [CLS81]. For example, a maximum clique in a cograph can be identified from a parse tree representation in parallel at the cost of tree contraction.

Acknowledgment. We wish to express our thanks to Nicholas Pippenger who first pointed out the connection between our work on parallel processing of region search trees [DK87] and Miller and Reif's work on parallel tree contraction [MR85].

References

[ADKP87a] K. Abrahamson, N. Dadoun, D. Kirkpatrick and T. Przytycka. "A simple optimal randomized parallel list ranking algorithm". Computer Science Department Technical Report 87-14, University of British Columbia, Vancouver, May 1987.

[ADKP87b] K. Abrahamson, N. Dadoun, D. Kirkpatrick and T. Przytycka. "A simple parallel tree contraction algorithm". Computer Science Department Technical Report 87-30, University of British Columbia, Vancouver, August 1987.

[B74] R. Brent. "The parallel evaluation of general arithmetic expressions". *Journal of the ACM* 21, 2, April 1974, pp. 201-206.

[BV85] I. Bar-On and U. Vishkin. "Optimal parallel generation of a computation tree form". *ACM Transactions on Programming Languages and Systems* 7,2, 1985, pp. 348-357.

[CLS81] D.G. Corneil, H. Lerchs and L. Stewart. "Complement reducible graphs. " *Journal of Discrete and Applied Mathematics* 3, 1981, pp. 163-175.

[CV86a] R. Cole and U. Vishkin. "Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. " In *18th Annual Symposium on Theory of Computing*, 1986, pp. 206-219.

[CV86b] R. Cole and U. Vishkin. "Approximate and exact parallel scheduling with applications to list, tree and graph problems. " In *27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 478-491.

[CV86c] R. Cole and U. Vishkin. "The accelerated centroid decomposition technique for optimal parallel tree evaluation in logarithmic time". Ultracomputer Note #108, TR-242, Dept. of Computer Science, Courant Institute NYU, 1986.

[DK87] N. Dadoun and D. Kirkpatrick. "Parallel processing for efficient subdivision search". In *3rd ACM Symposium on Computational Geometry*, Waterloo, Ontario, 1987, pp. 205-214.

[GR86] A. Gibbons and W. Rytter. "An optimal parallel algorithm for dynamic tree expression evaluation and its applications". In *Symp. on Foundations of Software Technology and Theoretical Comp. Sci.*, 1986, pp. 453-469.

[H86] X. He. "Efficient parallel algorithms for solving some tree problems. " In *24th Allerton Conference on Communication, Control and Computing*, 1986, pp. 777-786.

[K68] D. Knuth, *The Art of Computer Programming Volume 1: Fundamental Algorithms*, Addison-Wesley, 1968.

[M83] N. Megiddo. "Applying parallel computation algorithms in the design of serial algorithms. " *Journal of the ACM*, 30, 4, Oct. 1983, pp. 852-865.

[MR85] G. L. Miller and J. Reif. "Parallel tree contraction and its application. " In *26th IEEE Symp. on Foundations of Computer Science*, 1985, pp. 478-89.

[TV84] R. E. Tarjan and U. Vishkin. "Finding biconnected components and computing tree functions in logarithmic parallel time. " In *25th Annual Symp. on Foundations of Comp. Science*, 1984, pp. 12-22.

[V83] U. Vishkin. "Synchronous parallel computation — a survey. " TR-71, Dept. of Computer Science, Courant Institute, NYU, 1983.

AN EFFICIENT ASYNCHRONOUS DISTRIBUTED PROTOCOL TO TEST FEASIBILITY OF THE DUAL TRANSSHIPMENT PROBLEM

M. A. COMEAU
Centre de recherche informatique de Montréal
Montréal (Canada)

K. THULASIRAMAN
Dept. of Electrical and Computer Engineering
Concordia University, Montréal (Canada)

K. B. LAKSHMANAN
Dept. of Computer Science
Concordia University, Montréal (Canada)

ABSTRACT

An asynchronous distributed protocol to test feasibility of the dual transshipment problem is presented. The message and time complexities of this protocol are $O(mn)$ and $O(n)$, respectively, where m and n are the number of edges and the number of nodes in the graph underlying the problem. This protocol can be easily modified to construct a starting solution required to initialize the primal-dual method. A mechanism is incorporated in these protocols to detect infeasibility, whenever it occurs.

I. INTRODUCTION

Network optimization refers to the class of optimization problems defined on graphs and networks. In recent years, efficient distributed protocols or algorithms have been obtained for several network optimization problems such as the single-source shortest-path problem, the maximum-flow problem, the problem of constructing a maximum matching, etc. In this paper, we are concerned with certain issues in the design of distributed protocols for the transshipment problem [1] which generalizes several of the network optimization problems such as the ones stated above.

The transshipment problem can be formulated as a linear program and therefore can be solved by the simplex method using any one of the three approaches, namely, the primal, the dual or the primal-dual methods. The network simplex algorithm is an efficient implementation of the primal method for solving the transshipment problem and has been extensively studied in the literature. However, the dual method has received very little attention, though one can construct an optimum solution of the problem from an optimum solution of the dual. Our recent investigations [2] indicate that the network simplex algorithm is essentially sequential in nature and does not offer much scope for an efficient distributed implementation. This has motivated us to select the dual and the primal-dual methods as candidates in our study of issues in distributed protocol design for network optimization problems.

The simplex method involves two phases. In the first phase, feasibility of the given problem is tested and a basic feasible solution is constructed, if such a solution exists. The second phase starts with a basic feasible solution and through a sequence of pivot operations constructs an optimum solution, if available.

In this paper, we will be primarily concerned with the first phase of the dual transshipment problem. Thus, we are interested in designing an efficient distributed protocol to test feasibility of this problem. In Section II, we present the dual transshipment problem and present the important features of a sequential algorithm to test feasibility. This algorithm is of time complexity $O(mn)$ where m and n are, respectively, the number of edges and the number of nodes of the graph underlying the dual transshipment problem. In Section III, we show that the algorithm of Section II can be suitably modified to construct a starting solution required to initialize the primal-dual method. In Section IV, we present the model we use for distributed computation and then present, in Section V, the essential features of a synchronous distributed protocol for the feasibility testing problem. This protocol is of message complexity $O(mn)$ and time complexity $O(n)$. Combining this synchronous protocol with the α-synchronizer of Awerbuch [3],[4] or with the simpler synchronizer presented in [10] leads to an asynchronous protocol whose message and time complexities are the same as those of the synchronous protocol.

This research was supported in part by Concordia University under grant CASA-N67 and by FCAR Québec under grant 87AS2407 of the "Actions Spontanées" program.

II. ALGORITHM FEASIBLE: TESTING FEASIBILITY OF THE DUAL TRANSSHIPMENT PROBLEM

Given a weighted graph $G = (V,E)$ on n nodes and m edges, with the weight vector M_0 associated with the edge set E, let A denote the incidence matrix of G. Then the ***dual transshipment*** problem is a linear program defined as follows:

$$\textit{minimize } \Omega\Sigma$$

$$\textit{subject to } A^t\Sigma \geq -M_0 \tag{1}$$

$$\Sigma \geq 0 \tag{2}$$

where Σ is a column vector of dimension n and Ω is a row vector also of dimension n. Vectors Ω and M_0 are specified.

A vector $\Sigma \geq 0$ which satisfies (1) is called a ***feasible*** solution of the dual transshipment problem. If such a vector exists, then the given dual transshipment problem is feasible.

Theorem 1:
The dual transshipment problem is feasible if and only if the weighted graph G has no directed circuit of negative weight. //

Let d_{ij} denote the weight of a minimum-weight directed path from node i to node j. Let

$$\gamma_i = \max\{0, -\min_j\{d_{ij}\}\}, \ i = 1,2,\cdots,n. \tag{3}$$

Theorem 2:
The vector $\Sigma = (\gamma_1,\gamma_2,\cdots,\gamma_n)^t$ is a feasible solution of the dual transshipment problem if the weighted graph G has no directed circuit of negative weight. //

Proof of the above theorems may be found in [11].

To present an algorithm which obtains the feasible solution given in the above theorem, let us define the ***node firing*** operation as follows. Firing x times a node v refers to the operation of adding x to the weight of every outgoing edge at v and subtracting x from the weight of every incoming edge at v. In the following, $M(e)$ denotes the weight of edge e.

Algorithm FEASIBLE:

```
1. Let $M = M_0$.
2.While there exists an edge $e = (i,j)$ such that $M(e) < 0$, fire node $i$, $-M(e)$ times, updating $M$. //
```

Theorem 3:
If the weighted graph has no negative-weight directed circuits under the weight vector M_0, then Algorithm **FEASIBLE** terminates in a finite number of steps after firing each node i exactly γ_i times. //

Proof of correctness and termination of this algorithm may be found in [11].

We next present an implementation of the above algorithm which achieves a time complexity of $O(mn)$. The main steps in this implementation are:

Step 1: For each i, set $\gamma_i = 0$. Let M be the initial weight vector associated with the edge set.

Step 2: For each i, compute

$$\sigma_i = \max\{0, -\min_j\{M(i,j)\}\}$$

Step 3: Fire each node i, σ_i times and update γ_i by adding σ_i to its current value (Note that firing results in updating M also).

Step 4: If $\sigma_i = 0$ for all i, STOP. ELSE return to Step 2.

We now proceed to prove that the above implementation of Algorithm **FEASIBLE** achieves a complexity of $O(mn)$. Our proof also shows that this implementation computes γ_i's correctly whenever there is no circuit of negative weight in G.

In the following, we shall refer to one execution of Step 2 and Step 3 as a ***sweep***. Also, whenever there is no negative-weight directed circuits, it can be proved that the graph has at least one node i with $\gamma_i = 0$. Such a node will be referred to as a ***datum*** node. Note that in a graph, there may be more than one datum node. It can be proved that Algorithm **FEASIBLE** will never fire a datum node. A directed path of weight $-\gamma_i$ will be denoted by $p_{i,i'}$ where i' is clearly a datum node. We refer to the number of edges in a path as the ***length*** of the path.

Theorem 4:

Assume that G has no negative-weight directed circuits under the initial weight vector M_0. Consider any node i for which $\gamma_i > 0$. If the length of the path $p_{ii'}$ is k, then the node i will have been fired γ_i times at the end of the k^{th} sweep in the above implementation of **FEASIBLE**.

Proof:

Let $p_{ii'} = i, i_1, i_2, \cdots, i_{k-1}, i'$. At the beginning of the first sweep, the weight on the edge (i_{k-1}, i') is $-\gamma_{ik-1}$. Also. i' is never fired. So at the end of the first sweep, node i_{k-1} will have been fired γ_{ik-1} times. At the beginning of the second sweep, i_{k-1} is a datum in the new weight pattern and the weight on the edge (i_{k-2}, i_{k-1}) will be $-\gamma_{ik-2}$. So, during this sweep, node i_{k-2} will be fired γ_{ik-2} times. Repeating these arguments, we can see that at the end of the k^{th} sweep, node i will have been fired γ_i times. //

Theorem 5:

If G has no negative-weight directed circuits under M_0, then Algorithm **FEASIBLE** will terminate in no more than n sweeps, where n is the number of nodes in G.

Proof:

Each directed path $p_{ii'}$ in G is of length $\leq n$-1. So, by Theorem 4, each node i will be fired a total of γ_i times in no more than n-1 sweeps and the theorem follows. //

During each sweep, $O(m)$ edges are examined. Thus, each sweep takes $O(m)$ time and, hence, we have the following theorem.

Theorem 6:

The complexity of Algorithm **FEASIBLE** is $O(mn)$ if G has no negative-weight directed circuits under the initial weight pattern M_0. //

Any $\sigma_i > 0$ after the n^{th} sweep of Algorithm **FEASIBLE** indicates the presence of a negative-weight directed circuit. We can also incorporate in this algorithm a mechanism to detect the edges in such a circuit. Algorithm **FEASIBLE** can be suitably modified to handle the general case where there are upper as well as lowerbounds specified on the firing numbers.

III. INITIALIZATION OF THE PRIMAL-DUAL METHOD

The ***upperbounded*** transshipment problem is as follows:

$$\begin{aligned} &\textit{minimize } cx \\ &\textit{subject to } Ax = b,\ 0 \leq x \leq u. \end{aligned} \tag{4}$$

To initialize the primal-dual method, one is required to determine a set of x_{ij}'s and γ_{ij}'s such that

1. $0 \leq x_{ij} \leq u_{ij}$ (5)

2. $x_{ij} = 0$ whenever $\gamma_i + c_{ij} \geq \gamma_j$ (6)

3. $x_{ij} = u_{ij}$ whenever $\gamma_i + c_{ij} < \gamma_j$ (7)

It can be seen [2] that the initialization problem reduces to finding γ_i's such that

$$\gamma_i - \gamma_j + c_{ij} \geq 0 \tag{8}$$

whenever $u_{ij} = \infty$. If such γ_i's can be found then we can set

$$x_{ij} = \begin{cases} 0 & \text{if } \gamma_i + c_{ij} \geq \gamma_j \\ u_{ij} & \text{if } \gamma_i + c_{ij} < \gamma_j. \end{cases} \tag{9}$$

Suppose we modify Algorithm **FEASIBLE** so that edges for which $u_{ij} < \infty$ are ignored while computing σ_i. Then we can see that such a modified algorithm will compute correctly a set of γ_i's (whenever such a set exists) satisfying (9).

IV. MODEL OF DISTRIBUTED COMPUTATION

For synchronous computations, we follow the model used in [3] - [5] and for asynchronous computations, we follow the model used in [3], [4], [6] - [9]. These models are by far the most commonly used ones. It might be pointed out that in these models it is necessary that all the messages received at a processor are transferred to a single common queue before being processed one by one. It is also assumed that the actions necessary for processing a message can all be performed in negligible computation time, without wait once started, and also uninterrupted by the arrival of other messages. Note also that in the asynchronous model, the communication subsystem is assumed to deliver a message to its destination after *a finite but, undetermined time lapse*. On the other hand, in a synchronous network model, we assume the existence of a global clock, so that all messages are sent only when a clock pulse is generated. Moreover, a message sent by any processor to its neighbor arrives at its destination before the next pulse is generated.

It must be emphasized that in all our protocols, no processor is assumed to be aware of the number of processors in the system.

As regards complexities, the message complexity of a distributed protocol is the total number of messages transmitted during the execution of the algorithm. The time complexity (both in the synchronous and the asynchronous cases) is the time that elapses from the beginning to the termination of the algorithm, assuming that the delay in any link is exactly one unit of time. Note that this assumption of unit delay in links, even for the asynchronous case, is made only for the purpose of timing analysis.

V. A SYNCHRONOUS DISTRIBUTED PROTOCOL

We now present the essential features of a synchronous distributed protocol for testing feasibility of the dual transshipment problem. This protocol is given in two phases. In the following, we shall assume that each node processor v is aware of the weights of all the outgoing edges at v.

After the usual **WAKE-UP** protocol, a leader s is first elected and a tree T rooted at s is constructed. The root s then informs all the nodes to start Phase 1. (Note that leader election requires $O(m + n \log n)$ messages and $O(n)$ time [12]).

Phase 1:

To start with, the firing number γ_w of each node w is set to zero.

Two types of messages -WAVE and ACK- are used. During the first pulse of activity each node processor w sets itself to the **START** state, sets PRED(w) = w and computes the minimum of the weights of all the outgoing edges incident at$w$. Let this minimum be ∂_w. If $\partial_w > 0$ then node w changes its state to

SUCCESS. Otherwise, it updates the weight of each outgoing edge by adding $|\partial_w|$ to its current weight and updates its firing number by adding $|\partial_w|$ to the current value of γ_w. Also, a WAVE message carrying the value $|\partial_w|$ is broadcast along each incoming edge at w.

During each subsequent pulse the node w processes the messages received, one at a time. Consider the WAVE message received along the i^{th} outgoing edge. Let FIRE(i) denote the value carried by the message. Now node w updates the weight of the i^{th} outgoing edge by adding -FIRE(i) to its current weight. Let MIN(w) denote the minimum of the current weights of all the outgoing edges at w after all the WAVE messages have been processed.

If MIN(w) ≥ 0, then node w sends an ACK message to every neighbor from which a WAVE message has been received. If MIN(w) < 0, then node w updates its firing number by adding |MIN(w)| to its current value. The weight of every outgoing edge is updated by adding |MIN(w)| to the current weight and then an ACK message is sent to PRED(w) if PRED(w) $\neq w$. Node w then arbitrarily selects an edge, say the edge (w,k) whose current weight is zero, sets PRED(w) = k, and sends ACK messages for all the WAVE messages processed, except for the one which was received along the edge (w,k). Finally, a WAVE message which carries the value of |MIN(w)| is sent along each incoming edge. If no incoming edge is present at w, an ACK message will be sent to PRED(w).

Note that at any time an ACK message will be pending on no more than one outgoing edge incident at any node. When node w has received ACK messages for all the messages it has sent, it transmits an ACK message to PRED(w).

Node w sets its state to **SUCCESS** when it has received ACK messages for all the messages it has sent during its first pulse.

When a node w and all its descendants in the tree T have reached the **SUCCESS** state, node w informs its father in T accordingly. When the leader s recognizes that all the nodes including itself are in the **SUCCESS** state, it initiates Phase 2.

Phase 2:

The objective of Phase 2 is to test for the presence of a directed circuit of negative weight. This is done by the leader transmitting messages along T and checking if at any node an ACK message is pending on an outgoing edge. If so, it indicates the presence of a negative-weight directed circuit, and all the nodes are then switched to the **INFEASIBLE** state (indicating infeasibility of the problem.) Otherwise, all the nodes are switched to the **FINISHED** state; in this case the current firing numbers of the nodes give a feasible solution for the given dual transshipment problem.

VI. CORRECTNESS AND COMPLEXITY OF THE SYNCHRONOUS PROTOCOL

To prove the correctness and establish the complexity of the synchronous protocol, first note that the actions taken by the nodes during any pulse of the first phase are essentially the same as those during the corresponding sweep of the sequential Algorithm **FEASIBLE**. Thus, if we assume that there are no negative-weight directed circuits, then the firing numbers of all the nodes would reach the values as given by (3) in no more than n pulses and hence, the weights of all edges would attain nonnegative values within this period. No WAVE messages will flow in the graph after the n^{th} pulse. The crucial problem then is to prove the correctness of the mechanism we have used to let every node know that all the edges have attained nonnegative weights and that the objective of the protocol has been achieved.

To do so, let us first assume that there are no negative weight directed circuits and that all the edges with negative weights are incident out of a single node w. Let the edge having the most negative weight be (w,i). Thus, during the first pulse of the first phase, node w is the only one which initiates a WAVE message. Any node x initiating a WAVE message during the first pulse will be considered as initiating the x-wave. The purpose of the w-wave is to fire node w so that the weight of the edge (w,i) becomes zero. However, this firing may cause weights of some of the edges incident into node w to become negative and thus will, in turn, trigger new WAVE messages (in fact w-wave messages). Let G_w denote the set of all edges traversed by the w-wave messages.

Note that a WAVE message received at, say, node j, along edge (j,k) is acknowledged either i) when it triggers no new WAVE messages or ii) when all the WAVE messages it triggered have been acknowledged or iii) when a later WAVE message arriving at j causes further firing of node j. In all these cases, all the edges traversed by the w-wave message received at node j along edge (j,k) would have attained nonnegative values. This means that when node w has received ACK messages for all the messages it sent during the first pulse, all edges in G_w would have attained nonnegative values. Therefore, at this point, node w detects completion of the mission of the w-wave message and sets itself to the **SUCCESS** state.

In the general case, more than one node may initiate a wave during the first pulse. The subgraphs traversed by the waves may not be distinct. When all the edges in any one of these subgraphs have attained nonnegative weights, the node initiating the corresponding wave would set itself to the **SUCCESS** state. When all the nodes reach the **SUCCESS** state, the root node can detect this and in the second phase it will inform all the nodes about the successful termination of the protocol.

Consider next, the case when there are some negative directed circuits. Suppose a node, say j, in such a circuit sends a WAVE message along edge (k,j) during the first pulse. Since j is in a negative-weight directed circuit and the length of a circuit is at most n, this node will be fired again within the first n pulses, resulting in a WAVE message along (k,j). This, in turn, sends an ACK message for the message sent along (k,j) during the first pulse and a new ACK message will be pending on (k,j). Thus node j will be able to set itself to the **SUCCESS** state within the first n pulses. The new pending ACK message on (k,j) indicates the presence of a negative-weight directed circuit and will be detected during the second phase. It can be shown that an ACK message will be pending at at least one node in a negative-weight directed circuit.

It is clear from the above discussion that all nodes will reach the **SUCCESS** state within the first n pulses. The root will detect this in $O(n)$ pulses. The second phase will terminate in $O(n)$ pulses. Furthermore, the number of messages during each pulse is $O(m)$. As pointed out above, leader election can be accomplished using $O(m + n \log n)$ messages and $O(n)$ time. Thus, we have the following theorem.

Theorem 7:
A feasible solution, if it exists, to a dual transshipment problem defined on a network of m edges and n nodes can be computed using a synchronous distributed protocol which requires $O(mn)$ messages and $O(n)$ time. //

VI. AN ASYNCHRONOUS DISTRIBUTED PROTOCOL

Combining the synchronous protocol of the previous section with the α-synchronizer of Awerbuch [3], [4] or the simpler synchronizer presented in [10], an asynchronous distributed protocol requiring $O(mn)$ messages and $O(n)$ time can be constructed for testing feasibility of the dual transshipment problem. Note that the synchronizer is required only for Phase 1 of this protocol.

VII. CONCLUSIONS

We have shown that an asynchronous distributed protocol of message complexity $O(mn)$ and time complexity $O(n)$ can be constructed to test feasibility of a dual transshipment problem defined on a network of m edges and n nodes.

VII. REFERENCES

[1] V. Chvátal, "Linear Programming", Freeman Company, San Francisco, 1983.

[2] Marc Comeau, "Reachability and Sequencing in Marked Graphs and State Graphs: Algorithms Based on Network Programming", Ph.D. thesis, Dept. of Electrical Engineering, Concordia University, June 1986.

[3] B. Awerbuch, "Complexity of Network Synchronization", J. Assoc. Comput. Mach., Vol. 32, No. 4, Oct. 1985, pp. 804-823.

[4] B. Awerbuch, "Reducing Complexities of the Distributed Max-Flow and Breadth-First-Search Algorithms by means of Network Synchronization", Networks, Vol. 15, 1985, pp. 425-437.

[5] E. Korach, D. Rotem and N. Santoro, "Distributed Algorithms for finding centers and medians in Networks", ACM Trans. Prog. Lang. Systems, Vol. 6, No. 3, July 1984, pp. 380-401.

[6] K. M. Chandy and J. Misra, "Distributed Computation on Graphs: Shortest-path Algorithms", Comm. Assoc. Comput. Mach. Vol. 25, No. 11, Nov. 1982, pp. 425-437.

[7] R. G. Gallager, "Distributed Minimum Hop Protocols", Technical Report LIDS-P-1175, Massachusetts Institute of Technology, U.S.A., Jan 1982.

[8] K. B. Lakshmanan, N. Meenakshi and K. Thulasiraman, "A Time-Optimal Message Efficient Distributed Algorithm for Depth-First-Search", Info. proc. Letters, Vol. 25, No. 2, May 1987.

[9] A. Segall, "Distributed Network Protocols", IEEE Trans. Info. Theory, Vol. IT-29, No. 1, Jan 1983, pp. 23-35.

[10] K. B. Lakshmanan and K. Thulasiraman, "On the Use of Synchronizers for Asynchronous Communication Networks", Proceedings of the 2^{nd} International Workshop on Distributed Algorithms, Amsterdam, July 1987.

[11] M. A. Comeau and K. Thulasiraman, "Structure of the Submarking Reachability Problem and Network Programming", To appear in the IEEE Transactions on Circuits and Systems.

[12] B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems", Proc. 19^{th} Annual ACM Symp. on Theory of Computing, New York City, May 1987.

UNIVERSAL RANKINGS FOR BROADCASTING IN TREE NETWORKS

Arnon Rosenthal
Computer Corporation
of America
4 Cambridge Center
Cambridge, Mass. 02142

Peter Scheuermann
Northwestern University
Dept. of Elect. Engineering
and Computer Science
Evanston, Il. 60201

ABSTRACT

We consider the problem of optimal broadcasting in a tree network which uses point-to-point communication. We define the properties of well-behaved cost measures and show that for these one can obtain a universal ranking, i.e. an ordering among the adjacent edges of each node which can be used to yield optimal broadcasting irrespective of the message originator. Efficient linear and quasilinear algorithms are presented to obtain the rankings and broadcast costs.

INTRODUCTION

We consider the problem of broadcasting a message to all of the nodes in a point-to-point computer network. A broadcast in a network with n nodes starts at a node y, called the originator, and consists of a series of n-1 calls that satisfy the following rules:

1) Each call requires one unit of time.
2) A node can only call an adjacent node.
3) A node can participate in only one call during a time unit.

Thus during a given time unit we can have a number of parallel calls but only one call involving a given node. After each node receives the message the broadcasting is considered complete. A related problem is the one refered to as polling [3], where after each node receives the message it has to return a response to the originator.

A broadcast is called optimal if it satisfies the chosen measure of broadcast cost. In [6] Slater et.al. present a linear algorithm for minimizing the maximum delay in a tree network for every possible originator. Heuristic algorithms for achieving efficient near-optimal broadcasts which also minimize the maximum delay in an arbitrary network were presented in [8]. The dual problem was considered in [4] : namely, given an integer n, how to construct a minimal broadcast graph, i.e. a graph with the smallest number of edges whose maximum delay equals that of the complete graph with n nodes.

Finding an optimal broadcast for a given originator is equivalent to providing each node u in the corresponding spanning tree with a routing table, i.e., an ordering of all its incident edges except the edge along which u received the message. The implicit assumption in all previous broadcasting algorithms [5,6,8] which considered the optimization of maximum delay was that for every originator it is necessary to provide each node in the network with a different ordering of edges. Obviously, for fairly large networks this requires a considerable amount of storage.

In this paper we define a number of "well behaved" cost measures for broadcasting and show that for these in tree networks it suffices to store at each node only one routing table, consisting of an ordered list of all its edges, which guarantees an optimal broadcast regardless of the identity of the originator. Furthermore, linear and quasilinear algorithms are presented to obtain the rankings of the neighbours of each node.

URGENCIES AND UNIVERSAL RANKINGS

Let T be a tree network and (u,v) an edge in T. We shall denote by T(u,v) and T(v,u) the subtrees of T consisting of the components of T - (u,v) containing u and v, respectively. In Figure 1 T(u,v) includes the edges in solid lines and the nodes incident to them. A **ranking** at each node v is an ordered list of its incident edges. A **full ranking** , denoted r(v), orders all the incident edges of v, while a ranking which omits the edge (v,v_0) is called **partial** and is denoted $r(v,v_0)$. Similarly, R will denote the set of full rankings for T, and $R(v,v_0)$ will stand for the set of rankings in $T(v,v_0)$. A set of rankings is called **universal** if, for every originator, it yields an optimal broadcast for a given cost measure. The message-forwarding protocol for each vertex v is the following. When v receives the message over an edge (u,v) it omits this edge from its full ranking r(v), to obtain the partial ranking r(v,u). The message is then forwaded to the remaining neighbors included in r(v,u), one neighbor at a time. Thus R(v,u) is used to direct the broadcast from v through T(v,u).

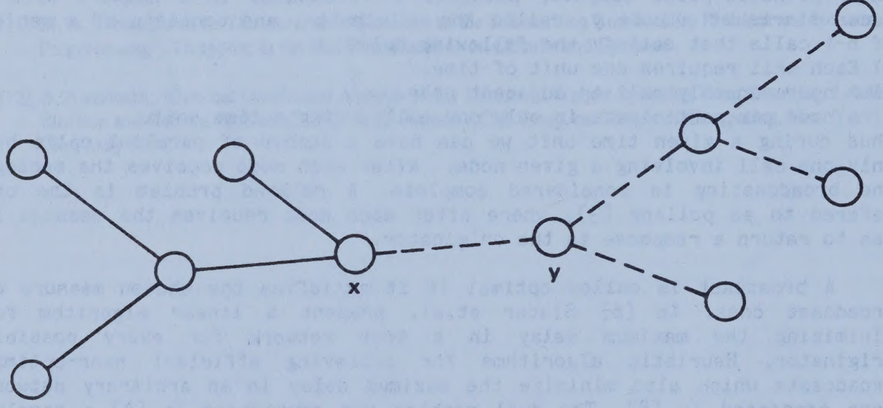


Figure 1

To obtain a uniform notation we assume the existence of a pseudonode o which is connected to the broadcast originator y. Then, T(y,o) will denote the tree rooted at y. Let $d_R(T(y,o),v)$ denote the time it takes for the call originating at y to reach v using the ranking R. We now define the following cost measures:

$$(1)\quad TD_R(y,o) = \Sigma_v\{\ d_R(T(y,o),v))\mid v\varepsilon V\ \}$$

$$(2)\quad MD(y,o) = \max_v\{\ d_R(T(y,o),v))\mid v\varepsilon V\ \}$$

where TD and MD denote the total delay and the maximum delay, respectively. Figure 2 below shows a sample tree network with a universal set of rankings for MD. The numbers on the edges next to a node v stand for r(v), while the number inside a node v stands for d(T(y,o),v).

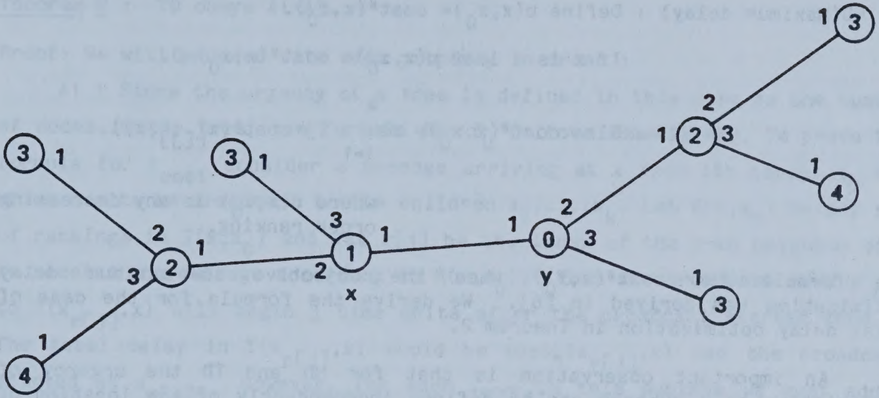


Figure 2

Although our algorithms are presented for the above cost measures, we abstract them slightly in order to show that our results can be applied to other cost measures also. Thus we shall denote by $\mathrm{cost}_R(x,z)$ the cost of broadcasting in the subtree $T(x,z)$ using the rankings $R(x,z)$ and we use $\mathrm{cost}^*(x,z)$ if $R(x,z)$ is optimal.

For the above cost measures it is also possible to define a measure of **urgency** of reaching the root of $T(x,x_0)$, under the assumption that the broadcast in the subtree $T(x,x_0)$ will then be handled optimally. This urgency measure will be denoted $u(x,x_0)$. The neighbors of a node x are referred as $x_0, x_1, \ldots, x_k$. Node x_0 always denotes the neighbor on the path to o, and is called the parent; $x_1,\ldots,x_k$ are called the children of x in $T(x,x_0)$. Now, $x_{r[j]}$ will stand for the j-th child of x to receive the message according to ranking r. Correspondingly, a decreasing-urgency ranking $r(x)$ is one that orders the edges in nonincreasing order of urgency for $T(x_i,x)$, with ties broken arbitrarily. We can now state the equations for deriving the urgencies and cost functions for TD and MD:

TD(Total delay) : Define $u(x,x_0)$ to be the number of nodes in $T(x,x_0)$
If x is a leaf : $u(x,x_0)=1$; $\mathrm{cost}^*(x,x_0)=0$.

$$\text{Else} \quad u(x,x_0)= 1+ \sum_{i=1}^{k} \{u(x_i,x)\};$$

$$\mathrm{cost}^*(x,x_0)= \sum_{j=1}^{k} \{j*u(x_{r[j]},x)+\mathrm{cost}^*(x_{r[j]},x)\}.$$

where $r(x,x_0)$ is any decreasing order ranking.

MD(Maximum delay) : Define $u(x,x_0) = \mathrm{cost}^*(x,x_0)$.

If x is a leaf $u(x,x_0) = \mathrm{cost}^*(x,x_0) = 0$.

$$\text{Else } \mathrm{cost}^*(x,x_0) = \max_{j=1}^{k} \{ j + \mathrm{cost}^*(x_{r[j]},x) \}.$$

where $r(x,x_0)$ is any decreasing order ranking.

The formula for $\mathrm{cost}^*(x,x_0)$ when the objective is maximum delay optimization was derived in [6]. We derive the formula for the case of total delay optimization in Theorem 2.

An important observation is that for MD and TD the urgency of transmitting to a subtree can be defined independently of the location of the originator. This is due to the fact that the underlying cost measures satisfy the following properties of being **well behaved** :

A1. Urgencies and optimal costs can be computed recursively.
A2. Rankings which are optimal for a subtree $T(x,z)$ considered in isolation remain optimal in the whole tree T.
A3. Any decreasing-urgency ranking $R(x,z)$ is optimal in $T(x,z)$.

We now state these properties more formally, so that they can be verified for TD and MD.

A1. Let $T(x,x_0)$ be any subtree and assume that each subtree $T(x_i,x)$ $T(x,x_0)$ has been optimally ranked $(i=1,\ldots,k)$. Let $R(x,x_0)$ denote the set of rankings obtained by using the optimal rankings on the subtrees $T(x_i,x)$, and ranking $r(x,x_0)$ at x. Let C^* and U denote the vectors of optimal costs and urgencies $[\mathrm{cost}^*(x_i,x)]$ and $[u(x_i,x)]$. Then there exist functions f_{cost} and f_u such that:

(1) $\mathrm{cost}_R(x,x_0) = f_{cost}[r(x,x_0),C^*,U]$ and

(2) $u(x,x_0) = f_u(C^*,U)$.

A2. Let $R(x.x_0)$ be any ranking of $T(x,x_0)$, and let $T' \subseteq T(x,x_0)$. Suppose $R'(x,x_0)$ is optimal for T' and identical to R at vertices outside T'. Then $\mathrm{cost}_{R'}(x,x_0) \leq \mathrm{cost}_R(x,x_0)$.

A3. In formula (1) of A1, f_{cost} is minimized by anu decreasing-urgency ranking.

The following observations follow immediately from properties A1 and A2 respectively:

Observation 1 : Costs and urgencies remain unchanged if two subtrees of equal urgency are interchanged in the ranking $R(x,x_0)$.

Observation 2 : $\mathrm{cost}^*(x,x_0) = \mathrm{Min}_{[\text{rankings } r(x,x_0)]} f_{cost}[r(x,x_0),C^*,U]$.

Theorem 1 : MD obeys properties A1-A3.
The proof is given in [6].

Theorem 2 : TD obeys A1-A3.

Proof: We will proceed with each property in order :

A1 : Since the urgency of a tree is defined in this case as the number of nodes in the tree, the formula for f_u follows immediately. To prove the formula for f_{cost} consider a message arriving at x from its parent x_0 and which has to be broadcast to its children $x_1,...,x_k$. Let $R(x,x_0)$ be any set of rankings in $T(x,x_0)$ and let r[j] be the index of the j-th neighbor of x to receive the message according to $R(x,x_0)$. Then the broadcast from $x_{r[j]}$ to $T(x_{r[j]},x)$ will begin j time units after the broadcast started from x. The total delay in $T(x_{r[j]},x)$ would be $cost_R(x_{r[j]},x)$ had the broadcast started at $x_{r[j]}$. However, for each node in this subtree we must add a contribution of j to TD since it takes j units to reach the root of the subtreee. Hence, the contribution of $T(x_{r[j]},x)$ to $TD_R(x,x_0)$ is $cost_R(x_{r[j]},x)+j*u(x_{r[j]},x)$. Summing over all subtrees we obtain the formula:

$$cost_R(x,x_0) = \Sigma_{j=1}^{k} \{j*u(x_{r[j]},x) + cost_R(x_{r[j]},x)\}. \quad (*)$$

If R is optimal on every subtree $T(x_{r[i]},x)$, then $cost_R(x_{r[j]},x) = cost^*(x_{r[j]},x)$, so C* may be used.

A2: This property may be proved by induction on subtree height. We omit the details.

A3: By observation 1, all decreasing-urgency rankings yield the same value of f_{cost}, so we need only show that for TD some decreasing urgency ranking minimizes f_{cost}. Consider any partial ranking $r(x,x_0)$ which is not decreasing-urgency, i.e. for some i, $u(x_{r[i]},x) < u(x_{r[i+1]},x)$. Produce $r'(x,x_0)$ by interchanging the subtrees in positions i and i+1, i.e. r'[i]=r[i+1] and r'[i+1]=r[i]. Let $R(x,x_0)$ be the set of rankings obtained by using $r(x,x_0)$ and $R(x_{r[i]},x)$, the optimal sets of rankings on the subtrees $T(x_{r[i]},x)$ (i=1,..,k). Let $R'(x,x_0)$ be identical to $R(x,x_0)$ except that $r'(x,x_0)$ replaces $r(x,x_0)$. Then from the formula (*) above we obtain: $cost_{R'}(x,x_0)-cost_R(x,x_0)=[i*u(x_{r[i+1]},x)+(i+1)*u(x_{r[i]},x)] - [i*u(x_{r[i]},x)+(i+1)*u(x_{r[i+1]},x)]= u(x_{r[i]},x)-u(x_{r[i+1]},x)<0$. Hence, $R(x,x_0)$ could not minimize f_{cost}, so only a decreasing-urgency ordering can minimize f_{cost}.

Theorem 3 : Any set of full, decreasing-urgency rankings is universal.

Proof: We first show by induction on the tree height that if R(v,o) is a decreasing-urgency ranking, it yields an optimal broadcast for T(v,o). Consider a message originating at any node w. Our protocol will cause the message to be forwarded to each node in T(w,o) according to decreasing-urgency ranking R(w,o), i.e. optimally.

ALGORTHMS FOR COMPUTING UNIVERSAL RANKINGS

We proceed now to present a generalized algorithm which will efficiently calculate subtree urgencies and produce the set of rankings R for any cost measure obeying properties A1-A3. The novelty of our algorithm compared to those presented in [6,7] is that it computes urgencies and optimal costs for a number of cost functions and it produces a universal ranking, versus a partial ranking which results in the other algorithms. It is important that the algorithm be efficient, since rerouting after link failures may require computing new rankings; one wishes to minimize the time until the network is again prepared to handle broadcsts.

The algorithm consists of two phases. We first choose an arbitrary vertex, denoted by root, as the root for a postorder traversal which computes R*(root,o) and cost*(root,o). Observe that R*(root,o) consists of a full ranking r(root) and partial rankings for x_i in T(root,o). The second phase consists of a preorder traversal of T(root,o) which produces a universal set of rankings and also computes cost*(v,o) for every v if so desired (note that on the initial call of Walk-down the insertion step is skipped, i.e. there is no need to insert u(o,root) into the full ranking of root).

```
Algorithm 1 : Post-ord(root,o);
               Walk-down(root,o);

       Proc Post-ord($x,x_0$)
       /* Computes $R^*(x,x_0)$, $cost^*(x,x_0)$ and $u(x,x_0)$ */
       If x is a leaf node then determine $u(x,x_0)$ from definition TD
      or MD
      Else do
             For each child $x_i$ do Post-ord($x_i$,x) od ;
             Sort $\{u(x_i,x)\}$ for i=1,...,k ;
             Let $r^*(x,x_0)$ rank the subtrees $T(x_i,x)$ in order of
             decreasing urgency ;
             Compute $cost^*(x,x_0)$ and $u(x,x_0)$ using definition of
             TD(Total delay) or MD(Maximum delay) ;
           od
     End Post-ord
      Proc Walk-down($x,x_0$)
      /* Traverses T in preorder determining the full ranking R */
      If $x_0 \neq o$ then insert $u(x,x_0)$ into the decreasing urgency
             partial ranking $r^*(x,x_0)$ to produce $r^*(x)$ ;
      For each child $x_i$ of x
       do  Compute $u(x,x_i)$ from $r^*(x)$ and $\{u(x_j,x) \mid j=0,1...,k; j \neq i\}$
            [Compute $cost^*(x,x_i)$ if so desired; ]
           Walk-down($x_i$,x);
         od
     End Walk-down
```

The details of the **Compute** step in phase two of Algorithm 1 are covered in our discussion below concerning efficient implementations for MD and TD.

We shall show first that Algorithm 1 can be implemented to run in linear time if tBhe objective is maximum delay, i.e. MD, minimization. If at vertex v we spend time bounded by c*degree(v) for some constant c, the entire algorithm will take $\Sigma_v c^*\mathrm{degree}(v) = c^*2^*(\text{number of edges}) = c'^*(\text{number of nodes})$, i.e. linear time. In other words, to achieve overall linear time, we must spend only a fixed amount of time at each neighbor of v.

In the implementation of Post-ord, the one difficulty we encounter regards the sorting of $\{ u(x_i,x) \mid i=1,\dots,k\}$. However, notice that if we compute first $w^* = \max_{i=1}^{k} u(x_i,x)$, then we know that only subtrees with delay of at least w^*-k can possibly supply the constraining delay. We may thus use a bin sort [1], employing k+1 bins to sort integers between w^*-k and w^* ; subtrees with urgency less then w^*-k may be ranked in arbitrary order at the end. Each child of x, i.e. each x_i, is put once in a corresponding bin and then taken out once to produce the ranking $r^*(x,x_0)$.

As far as Walk-down is concerned, it is clear that the insert step which produces the full ranking $r^*(x)$ from the partial ranking $r^*(x,x_0)$ can executed in O(degree(x)) time. It remains to show that we can **compute** each urgency $u(x,x_i)$ in a fixed amount of time. Let m be the index of the neighbor x_i which produced the maximum value in the calculation of $u(x,x_0) = \mathrm{cost}^*(x,x_0)$. Then we can derive immediately the formula:

$$u(x,x_i) = \begin{cases} u(x,x_0)-1 & \text{if } x_i = x_{r^*[k]} \text{ and } k<m \\ u(x,x_0) & \text{otherwise} \end{cases}$$

Thus a linear sweep through the list corresponding to $r^*(x)$ suffices to produce all the $u(x,x_i)$.

If Algorithm 1 is to be implemented for minimizing the total delay, i.e. TD, the Walk-down phase can still run in linear time. We observe first that by the definition of urgency in this case we obtain :

$$u(x,x_i) = u(x,x_0)-u(x_i,x).$$

The optimal costs $\mathrm{cost}^*(x,x_i)$ are no longer identical to urgencies, as was the case in the TD minimization. Let $x_{r[k]}$ denote the last child of x according to the partial ranking $r^*(x,x_0)$. From the definition of the optimal costs we obtain:

$$\mathrm{cost}^*(x,x_{r[k}]) = \mathrm{cost}^*(x,x_0)-\mathrm{cost}^*(x_{r[k]},x)-k^*u(x_{r[k]},x).$$

The first term on the right side, $\mathrm{cost}^*(x,x_0)$, is known from the postorder evaluation phase. The other optimal costs can be computed using the following recurrence relations, i.e. using a fixed amount of time:

$$\mathrm{cost}^*(x,x_{r[i]}) = \mathrm{cost}^*(x,x_{r[i+1]}) + [\ \mathrm{cost}^*(x_{r[i+1]},x)- \mathrm{cost}^*(x_{r[i]},x)]+i^*[u(x_{r[i+1]},x)-u(x_{r[i]},x)].$$

However, it does not appear possible to make the Post-ord phase of Algorithm 1 tun in linear time for TD. The algorithm will require a running time of O(n*(maximum vertex degree)), which we call **quasilinear.**

Algorithm 2 which we present below can be applied only for TD, but it has the advantage of running in linear time. It computes all the urgencies and the optimal set of rankings R. If delays as well as rankings are needed, we can call Walk-down(root,o) for some arbitrary root.

```
Algorithm 2 :  Walk-up(root,o)
                Bin-sort(U,R)

     Proc Walk-up(x,$x_0$)
     /* Calculates all the urgencies in T */
     If x is a leaf node then $u(x,x_0)=1$
    Else do
            For each child $x_i$ do Walk-up($x_i$,x) od ;
            $u(x,x_0)= 1+\mathrm{sum}\ u(x,x_i)$ ;
         od
    $u(x_0,x)= n-u(x,x_0)$ ;
    End Walk-up

     Proc Bin-Sort(U,R)
     /* U denotes the set of urgencies, while R stands for the set
        rankings produced   */
     We perform a bin sort  on the urgencies of all nodes
       simultaneously ;
    The sorted list of bins is traversed in descending order
       and whenever an edge (u,v) is encountered we place it
       in the next position in r(u) ;
    End Bin-Sort
```

In summary, we have shown that universal rankings can be used to provide optimal broadcasting in tree networks with very little storage overhead. This capability is quite useful in supporting a number of functions in local networks, such as the transfer of control messages required for synchronization or the update of a replicated file directory in a distributed database system [2].

REFERENCES

[1] A. Aho, J. Hopcroft and J. Ullman, Data Structures and Algorithms, Addisson-Wesley, 1983.

[2] S. Ceri and G. Pelagatti, Distributed Databases : Principles and Systems, McGraw-Hill, 1984.

[3] G. Cheston and S.T. Hedetniemi, "Polling in Tree Networks," Congressus Numerantium, Vol.41, pp.7-20, 1984.

[4] A.M.Fairley, "Minimum Broadcast Graphs," Networks, Vol. 9, pp. 313-332, 1979.

[5] A. Proskurowski, "Minimum Broadcast Trees," IEEE Trans. on Computers, Vol. C-30, pp. 363-366, May 1981.

[6] P.J. Slater, E.J. Cockayne and S.T. Hedetniemi, "Information Dissimination in Trees," SIAM Journal on Computing, Vol.10, No. 4, pp. 692-701, Nov. 1981.

[7] P. Scheuermann and M.Edelberg, "Optimal Broadcasting in Point-to-Point Networks," Tech. Report, Dept. of Electr. Eng. and Comput. Sci., Nortwestern University, 1981.

[8] P. Scheuermann and G. Wu, "Heuristic Algorithms for Broadcasting in Point-to-Point Networks," IEEE Trans. on Computers, Vol. C-33, No.9, September 1984.

MINIMIZING SCHEDULE LENGTH SUBJECT TO MINIMUM FLOW TIME

JOSEPH Y-T. LEUNG and GILBERT H. YOUNG
Computer Science Program
University of Texas at Dallas
Richardson, TX 75080

ABSTRACT

We consider the problem of scheduling n independent tasks on $m \geq 1$ identical processors with the objective of minimizing the schedule length subject to the constraint that it must have the minimum flow time. For nonpreemptive scheduling, it is known that the problem is NP-hard. We show in this paper that the problem is a lot easier for preemptive scheduling. Specifically, we give an $O(n \log n)$ time algorithm to find an optimal preemptive schedule. Our algorithm generates schedules with at most $m - 1$ preemptions. We also show that an optimal nonpreemptive schedule can be almost twice as long as an optimal preemptive schedule. Our results suggest that preemption is extremely beneficial in minimizing the flow time and the schedule length simultaneously.

1. INTRODUCTION.

In deterministic scheduling theory, schedule length and flow time are two commonly studied performance measures. Minimizing the schedule length has the effect of increasing the processor utilization, while minimizing the flow time tends to decrease the average number of unfinished tasks in the system. To obtain good overall performance, a system should aim at optimizing both performance measures. Unfortunately, the two criteria seem to conflict with each other. Scheduling algorithms that do well with respect to one measure tend to do poorly for the other. For example, the LPT rule [9] is known to produce schedules that are reasonably short, yet the schedules produced tend to have large flow time. On the other hand, the SPT rule [5] is known to be an optimal algorithm for minimizing the flow time, but it is usually outperformed by the LPT rule in producing short schedules. With the exception of [4,7,8], most of the scheduling algorithms studied in the literature have concentrated on just one of the two criteria. In this paper, we continue the work of [4,7,8] to study the scheduling problem of optimizing both performance measures simultaneously.

Formally, our problem can be stated as follows : We are given a set $TS = \{ T_1, T_2, \dots, T_n \}$ of n independent tasks with execution times $\{ p_1, p_2, \dots, p_n \}$ to be scheduled on $m \geq 1$ identical processors. If S is a schedule of TS on the m processors, then $f_i(S)$ denotes the finishing time of T_i in S. The schedule length of S, denoted by $SL(S)$, is defined to be $SL(S) = \max_{i=1}^{n} \{ f_i(S) \}$, and the flow time of S, denoted by $FT(S)$, is defined to be $FT(S) = \sum_{i=1}^{n} f_i(S)$. A minimum length schedule, S^+, is one such that $SL(S^+) \leq SL(S)$ for all schedules S, and a minimum flow time schedule, S^-, is one such that $FT(S^-) \leq FT(S)$ for all schedules S. Our goal is to find a schedule, S^*, that has the minimum schedule length among all minimum flow time schedules. That is, S^* is one such that $FT(S^*) = FT(S^-)$ and $SL(S^*) \leq SL(S^-)$ for all schedules S^-.

The set of tasks can be scheduled nonpreemptively or preemptively on the processors. In nonpreemptive scheduling, a task once begun execution must proceed to completion. In constrast, preemptive scheduling allows a task to be preempted, and resume execution later on a possibly different processor. It is assumed, however, that there are no time loss in preemption. We shall use the symbols S_{NP}^+ and S_P^+ to denote a minimum length nonpreemptive schedule and a minimum length preemptive schedule, respectively. Similarly, S_{NP}^-, S_P^-, S_{NP}^*, and S_P^* are defined analogously. It is known that finding a S_{NP}^+ is strongly NP-hard [6]. This problem, known as the multiprocessor scheduling problem or the makespan minimization problem in the literature, has been studied by many researchers [2,3,9,10,12,13,15]. Because of the intractability of the problem, most research efforts have been directed towards designing and analyzing fast approximation algorithms [2,9,10,12], or studying special cases of the problem which admit polynomial-time solutions [3,13,15]. Finding a S_P^+ is a lot easier [14]; it can be done in $O(n)$ time. In fact, the schedule length of S_P^+ is simply given by $SL(S_P^+) = \max \{ \max_{i=1}^{n} \{ p_i \}, (1/m) \sum_{i=1}^{n} p_i \}$. Recently, it has been shown [11] that for any given set of independent tasks to be scheduled on $m \geq 1$ identical processors, we have $SL(S_{NP}^+)/SL(S_P^+) \leq 2m/(m+1)$. Furthermore, the bound is achievable for every $m \geq 1$. Thus, when m gets large, a minimum length nonpreemptive schedule can be almost twice as long as a minimum length preemptive schedule. Hence, preemption can reduce the length of a schedule by a significant proportion.

Finding a S_{NP}^- can be done in $O(n \log n)$ time. The well-known SPT rule [5] always constructs a S_{NP}^- and it works as follows : Add enough zero-execution-time tasks so that n becomes an integer multiple of m. Sort the tasks in nondecreasing order of their execution times. Divide the tasks into $r = n/m$ ranks, with tasks $T_{(j-1)m+1}, T_{(j-1)m+2}, \ldots, T_{(j-1)m+m}$ in rank j, $1 \leq j \leq r$. Assign the tasks to the processors rank by rank, in order of increasing rank, each task in a given rank being assigned to a different processor. A schedule constructed by the SPT rule will be called a SPT schedule. Since there are $m!$ ways of assigning the tasks in any given rank, there are $(m!)^r$ SPT schedules for a set of rm tasks to be scheduled on m processors. Figure 1 shows two SPT schedules for a particular set of tasks. Observe that all SPT schedules give the same minimum flow time, $FT(S_{NP}^-) = \sum_{j=1}^{r} (r-j+1) \sum_{k=1}^{m} p_{(j-1)m+k}$. It has been shown [14] that preemption cannot reduce the minimum flow time. Thus, $FT(S_P^-) = FT(S_{NP}^-) = \sum_{j=1}^{r} (r-j+1) \sum_{k=1}^{m} p_{(j-1)m+k}$.

The problem of finding a S_{NP}^* has been shown to be NP-hard [1]. In [4], several fast approximation algorithms have been proposed and their worst-case performance bounds analyzed. In this paper, we give an $O(n \log n)$ time algorithm to find a S_P^*. Our algorithm generates schedules with at most $m - 1$ preemptions. The set of tasks given in Figure 2 shows that $SL(S_{NP}^*)/SL(S_P^*) = 2m/(m+1)$. From the result in [11], $2m/(m+1)$ is also an upper bound of $SL(S_{NP}^*)/SL(S_P^*)$ for any given set of tasks. Thus, preemption can be very beneficial in reducing the schedule length of a minimum flow time schedule.

Gonzalez [7] has given an $O(n \log n + nm)$ time algorithm to find a preemptive schedule with the minimum flow time subject to the constraint that the length of the schedule is no more than a given parameter, β. It is tempting to suggest that Gonzalez's algorithm can be used to find a S_P^* by repeatedly calling his algorithm with the parameter β, where β is the schedule length obtained in a binary search on an interval in which the minimum schedule length lies. However, the binary search may not terminate in a finite amount of time because the schedule length of S_P^* may be a real number even when the execution times of the tasks are integers. Moreoever, even if the number of iterations in the binary search can be bounded by a polynomial function of the size of the input, the overall running time of the algorithm will be substantially higher than our algorithm.

As will be seen in the next section, our algorithm employs a subroutine to preemptively schedule a subset of tasks to meet a certain deadline. This subroutine is actually a slight modification of Sahni's algorithm [16] which has been shown to solve the Deadline Scheduling problem. In the Deadline Scheduling problem, we are given $m \geq 1$ identical processors and a set of n independent tasks $TS = \{ T_1, T_2, \ldots, T_n \}$ with execution times $\{ p_1, p_2, \ldots, p_n \}$ and deadlines $\{ d_1, d_2, \ldots, d_n \}$. The problem is to determine if there is a preemptive schedule of TS on the m processors such that all deadlines are met. Such a schedule is called a feasible schedule. The idea of Sahni's algorithm is to schedule the tasks in nondecreasing order of their deadlines.

Assume the tasks have been indexed such that $d_1 \leq d_2 \leq \ldots \leq d_n$, and consider the scheduling of task T_i. Let $\mathrm{F}(k)$, $1 \leq k \leq m$, denote the finishing time of processor k after T_{i-1} has been scheduled. (Initially, $\mathrm{F}(k) = 0$ for all $1 \leq k \leq m$.) Let $k_1, k_2, \ldots, k_{x_i}$ be the indexes of the processors such that $\mathrm{F}(k_1) \leq \mathrm{F}(k_2) \leq \ldots \leq \mathrm{F}(k_{x_i}) < d_i$. The scheduling of T_i is done by one of the following four cases : (1) If $p_i > d_i - \mathrm{F}(k_1)$, then output "infeasible" (i.e., there are no feasible schedules for TS on m processors). (2) If $p_i \leq d_i - \mathrm{F}(k_{x_i})$, then schedule T_i completely on processor k_{x_i} and set $\mathrm{F}(k_{x_i})$ to be $\mathrm{F}(k_{x_i}) + p_i$. (3) If $p_i = d_i - \mathrm{F}(k_l)$ for some $1 \leq l < x_i$, then schedule T_i completely on processor k_l and set $\mathrm{F}(k_l)$ to be d_i. (4) If none of the above three cases apply, let processor k_l be such that $d_i - \mathrm{F}(k_{l+1}) < p_i < d_i - \mathrm{F}(k_l)$. Schedule $d_i - \mathrm{F}(k_{l+1})$ of T_i on processor k_{l+1} and the remainder of T_i on processor k_l. Set $\mathrm{F}(k_{l+1})$ to be d_i and $\mathrm{F}(k_l)$ to be $\mathrm{F}(k_l) + (p_i - (d_i - \mathrm{F}(k_{l+1})))$. The tasks are scheduled iteratively by the above procedure until either it outputs "infeasible", or all tasks have been scheduled. In the latter case, the schedule constructed by the algorithm is a feasible schedule.

We now define some conventions and notations that will be used consistently throughout the remainder of this paper. For a given set of n tasks, $TS = \{ T_1, T_2, \ldots, T_n \}$, to be scheduled on m processors, we always assume that the tasks have been indexed such that $p_1 \leq p_2 \leq \ldots \leq p_n$, and that $n = rm$ for some integer $r \geq 1$. For each $1 \leq j \leq r$, we call the tasks $T_{(j-1)m+1}, T_{(j-1)m+2}, \ldots, T_{(j-1)m+m}$ the rank j tasks. Furthermore, we say that rank(T_i) $= j$ if T_i is a rank j task. We define a particular SPT schedule, denoted by $SPT^{\#}$, as follows : For each $1 \leq k \leq m$, assign the tasks $T_k, T_{m+k}, \ldots, T_{(r-1)m+k}$ to processor k. It is easy to see that $SPT^{\#}$ has the longest schedule length among all SPT schedules for any given set of tasks. Furthermore, if $\mathrm{F}(k)$ denotes the finishing time of processor k in the $SPT^{\#}$ schedule, then we have $\mathrm{F}(k) = \sum_{j=1}^{r} p_{(j-1)m+k}$ for each $1 \leq k \leq m$. Clearly, we have $\mathrm{F}(k) \leq \mathrm{F}(k+1)$ for each $1 \leq k < m$. Finally, an optimal schedule always means a S_P^*.

The organization of the paper is as follows. In section 2, we describe our algorithm and analyze its running time. In section 3, we prove that our algorithm always constructs an optimal schedule for any given set of tasks. Finally, we draw some concluding remarks in the last section.

2. THE ALGORITHM.

In this section, we shall describe our algorithm and analyze its running time. Given a set of n tasks, $TS = \{ T_1, T_2, \ldots, T_n \}$, to be scheduled on m processors, our algorithm constructs a schedule as follows. First, we construct the $SPT^{\#}$ schedule for the tasks in the first $r - 1$ ranks. Next, we compute the optimal schedule length, denoted by OSL, using the finishing times of the processors in the $SPT^{\#}$ schedule and the execution times of the rank r tasks. As will be shown later, $\mathrm{OSL} = \max_{k=1}^{m} \{ \sum_{l=1}^{k} (\mathrm{F}(l) + p_{n-l+1})/k \}$, where $\mathrm{F}(l)$, $1 \leq l \leq m$, denotes the finishing time of processor l in the $SPT^{\#}$ schedule. Finally, we preemptively schedule the rank r tasks into the $SPT^{\#}$ schedule such that no two rank r tasks finish on the same processor, no processors have idle times before their finishing times, and no tasks finish later than OSL. Note that the two conditions that no two rank r tasks finish on the same processor and that no processors have idle times before their finishing times are to ensure that the schedule has the minimum flow time. The last step of our algorithm is done by a subroutine which is a modification of Sahni's algorithm. The rank r tasks are preemptively scheduled to meet the deadline OSL in the same manner as Sahni's procedure with the following three exceptions. First, Case 1 of Sahni's procedure no longer applies, since all tasks can meet the deadline OSL. Secondly, if a task T_i is scheduled on processor k_{x_i} by Case 2 of Sahni's procedure, then processor k_{x_i} will no longer be used to execute any subsequent tasks. This is to ensure that no two rank r tasks finish on the same processor. Finally, the tasks are scheduled in nonincreasing order of their execution times. A full description of our algorithm, to be called Algorithm B, is given below. A schedule produced by Algorithm B will be called a B-schedule.

Algorithm B

Input : A set $TS = \{ T_1, T_2, \dots , T_n \}$ of n tasks to be scheduled on $m \geq 1$ identical processors.

Output : An optimal schedule of TS on the m processors.

Method :

1. Construct the $SPT^{\#}$ schedule for the tasks $T_1, T_2, \dots , T_{(r-1)m}$.
2. $\mathrm{F}(k) \leftarrow \sum_{j=1}^{r-1} p_{(j-1)m+k}$ for each $1 \leq k \leq m$.
3. $\mathrm{OSL} \leftarrow \max_{k=1}^{m} \{ \sum_{l=1}^{k} (\mathrm{F}(l) + p_{n-l+1})/k \}$.
4. Initialize the set of available processors to be processors 1, 2, ... , m. Schedule the tasks $T_n, T_{n-1}, \dots , T_{n-m+1}$ in that order. Suppose we are scheduling task T_i. Let $k_1, k_2, \dots , k_{x_i}$ be the indexes of the available processors such that $\mathrm{F}(k_1) \leq \mathrm{F}(k_2) \leq \dots \leq \mathrm{F}(k_{x_i}) < \mathrm{OSL}$. T_i will be scheduled by one of the following three rules : (1) If $p_i \leq \mathrm{OSL} - \mathrm{F}(k_{x_i})$, then schedule T_i completely on processor k_{x_i}. $\mathrm{F}(k_{x_i}) \leftarrow \mathrm{F}(k_{x_i}) + p_i$. Delete processor k_{x_i} from the set of available processors. (2) If $p_i = \mathrm{OSL} - \mathrm{F}(k_l)$ for some $1 \leq l < x_i$, then schedule T_i completely on processor k_l. $\mathrm{F}(k_l) \leftarrow \mathrm{OSL}$. Delete processor k_l from the set of available processors. (3) If the above two cases do not apply, then let l be such that $\mathrm{OSL} - \mathrm{F}(k_{l+1}) < p_i < \mathrm{OSL} - \mathrm{F}(k_l)$. Schedule $\mathrm{OSL} - \mathrm{F}(k_{l+1})$ of T_i on processor k_{l+1} and the remainder of T_i on processor k_l. $\mathrm{F}(k_{l+1}) \leftarrow \mathrm{OSL}$. $\mathrm{F}(k_l) \leftarrow \mathrm{F}(k_l) + (p_i - (\mathrm{OSL} - \mathrm{F}(k_{l+1})))$. Delete processor k_{l+1} from the set of available processors. □

Shown in Figure 3 is a set of 12 tasks to be scheduled on 4 processors. Figure 3.a shows the $SPT^{\#}$ schedule constructed in Step 1 of the algorithm. The finishing times of processors 1, 2, 3, and 4 are 7, 9, 12, and 14, respectively. Step 3 of the algorithm gives OSL = max { (7+16)/1, (7+16+9+16)/2, (7+16+9+16+12+10)/3, (7+16+9+16+12+10+14+10)/4 } = 24. In Step 4, T_{12} is scheduled by rule (3) on processors 1 and 2. Then, processor 2 is deleted from the available set of processors. T_{11} is scheduled by rule (2) on processor 1 which is then deleted from the available set of processors. Finally, T_{10} and T_9 are both scheduled by rule (1) on processors 4 and 3, respectively.

The running time of Algorithm B is determined as follows. Steps 1 and 2 take $O(n)$ time. Step 3 takes $O(m)$ time. Step 4 can be implemented to run in $O(m \log m)$ time. Thus, the overall running time of Algorithm B is $O(n + m \log m)$ time. If we include the time needed to sort the tasks into nondecreasing order of their execution times, then the overall running time of the algorithm is $O(n \log n)$ time, since n is assumed to be larger than m. It is easy to see that a B-schedule has at most $m - 1$ preemptions.

3. ALGORITHM CORRECTNESS

In this section, we shall show that Algorithm B always constructs an optimal schedule. We first show that a B-schedule always gives the minimum flow time. Observe that in Step (4) of Algorithm B, the rank r tasks are scheduled in such a way that no two tasks finish on the same processor. This follows from the observation that if a task is scheduled by rules (1) or (2), then the processor to which it is assigned will not be used to schedule any subsequent tasks. If a task is scheduled by rule (3), then it finishes on processor k_{l+1} but not on processor k_l. Furthermore, processor k_{l+1} will not be used to schedule any subsequent tasks. Since a task is scheduled by one of the three rules, no two tasks can finish on the same processor. With this observation in mind, it is readily verified that the flow time of a B-schedule is $\sum_{j=1}^{r} (r-j+1) \sum_{k=1}^{m} p_{(j-1)m+k}$, which is the minimum as stated in section 1. For the remainder of this section, we shall concentrate on showing that a B-schedule has the minimum schedule length among all minimum flow time schedules.

Let S be a schedule of $TS = \{ T_1, T_2, \dots, T_n \}$ on m processors. With respect to S, we can define an ordering of the tasks $\rho(S) = (T_{i_1}, T_{i_2}, \dots, T_{i_n})$ such that $f_{i_1}(S) \le f_{i_2}(S) \le \dots \le f_{i_n}(S)$. We call this ordering the finishing time ordering of the tasks with respect to S. Furthermore, we call the tasks $T_{i_{(j-1)m+1}}, T_{i_{(j-1)m+2}}, \dots, T_{i_{(j-1)m+m}}$ the rank j tasks, $1 \le j \le r$, with respect to $\rho(S)$. A schedule S is said to be in canonical form if for each $1 \le k \le m$, the tasks $T_{i_k}, T_{i_{m+k}}, \dots, T_{i_{(r-1)m+k}}$ finish on processor k, where T_{i_l}, $1 \le l \le n$, is the l^{th} task in $\rho(S)$. The following theorem shows that every preemptive schedule can be converted into one in canonical form with no change in schedule length and flow time.

Theorem 1 : For every preemptive schedule S of TS on m processors, there is another schedule S' in canonical form such that $FT(S') = FT(S)$ and $SL(S') = SL(S)$.

Let S be a preemptive schedule in canonical form and let $\rho(S) = (T_{i_1}, T_{i_2}, \dots, T_{i_n})$ be the finishing time ordering of the tasks with respect to S. We divide S into r segments $E_1, E_2, \dots, E_r$ as follows. Segment E_1 consists of all the time intervals from time 0 until a rank 1 task (with respect to $\rho(S)$) finishes. That is, E_1 consists of the time intervals $[\, 0, f_{i_1}(S) \,]$ on processor 1, $[\, 0, f_{i_2}(S) \,]$ on processor 2, ..., $[\, 0, f_{i_m}(S) \,]$ on processor m. For each $2 \le j \le r$, E_j consists of the time intervals $[\, f_{i_{(j-2)m+1}}(S), f_{i_{(j-1)m+1}}(S) \,]$ on processor 1, $[\, f_{i_{(j-2)m+2}}(S), f_{i_{(j-1)m+2}}(S) \,]$ on processor 2, ..., $[\, f_{i_{(j-2)m+m}}(S), f_{i_{(j-1)m+m}}(S) \,]$ on processor m. We use ES_j to denote the total length of all time intervals in segment E_j. Thus, $ES_1 = \sum_{k=1}^{m} f_{i_k}(S)$, and for each $2 \le j \le r$, $ES_j = \sum_{k=1}^{m} (f_{i_{(j-1)m+k}}(S) - f_{i_{(j-2)m+k}}(S))$. With these definitions in mind, we can prove the following theorem which characterizes the structure of a minimum flow time preemptive schedule.

Theorem 2 : Let S be a minimum flow time preemptive schedule in canonical form, and let $\rho(S) = (T_{i_1}, T_{i_2}, \dots, T_{i_n})$ be the finishing time ordering of the tasks with respect to S. For each segment E_j, $1 \le j \le r$, we have : (1) there are no idle processor times in E_j, and (2) the tasks that are executed in E_j are rank j tasks (with respect to $\rho(S)$) only. Furthermore, every rank j task (with respect to $\rho(S)$) has execution time no larger than any rank $j+1$ task, $1 \le j < r$.

From Theorem 2, we see that a minimum flow time preemptive schedule, S, in canonical form is actually very similar to a B-schedule. The rank j tasks with respect to $\rho(S)$ are the same as the rank j tasks of the task system, $1 \le j \le r$. The tasks are executed rank by rank in S, and there are no idle processor times before the finishing time of a processor. If S is a schedule, we let $P(S)$ denote the partial schedule obtained from S by deleting all rank r tasks.

Lemma 1 : Let S be a minimum flow time preemptive schedule in canonical form and let S' be a B-schedule. Let $\mathrm{F}(k, S)$ and $\mathrm{F}(k, S')$ denote the finishing time of processor k in $P(S)$ and $P(S')$, respectively, $1 \le k \le m$. Then, we have $\sum_{k=l}^{m} (\mathrm{F}(k, S') - \mathrm{F}(k, S)) \ge 0$ for all $1 \le l \le m$.

Theorem 3 : Let S be a minimum flow time preemptive schedule in canonical form and let S' be a B-schedule. Then, there is another schedule $\hat{S}$ such that $FT(\hat{S}) = FT(S)$, $SL(\hat{S}) = SL(S)$, and $\mathrm{F}(k, \hat{S}) = \mathrm{F}(k, S')$, $1 \le k \le m$, where $\mathrm{F}(k, \hat{S})$ and $\mathrm{F}(k, S')$ denote the finishing time of processor k in $P(\hat{S})$ and $P(S')$, respectively.

By Theorems 1 and 3, an optimal schedule can always be found such that the tasks in the first r - 1 ranks are scheduled in the same manner as Algorithm B does. Thus, to show that a B-schedule has the minimum schedule length among all minimum flow time schedules, all we need to show is that Algorithm B schedules the rank r tasks in such a way that the resulting schedule length is minimum. In the following, we shall show that : (I) An optimal schedule must schedule the rank r tasks such that the resulting schedule length is at least OSL which is computed in Step 3 of Algorithm B, and (II) Algorithm B can always schedule the rank r tasks such that no rank r tasks finish later than OSL.

Before we show the above, we need to introduce the following notations. Let $F(k)$, $1 \leq k \leq m$, denote the finishing time of processor k of a partial schedule S such that $F(1) \leq F(2) \leq \ldots \leq F(m)$. The m-tuple PF = (F(1), F(2), ... , F(m)) is called the processor profile of S. For a given $D > F(m)$, the ordered pair (PF, D) is called a block and it consists of the time intervals [F(1), D] on processor 1, [F(2), D] on processor 2, ... , [F(m), D] on processor m.

Theorem 4 : Let PF = (F(1), F(2), ... , F(m)) be the processor profile of a partial schedule S, and let $TS' = \{ T_1', T_2', \ldots, T_m' \}$ be a set of m tasks such that $p_1' \geq p_2' \geq \ldots \geq p_m'$. Then, TS' can be preemptively scheduled into the block (PF, D) only if $D \geq \max_{k=1}^{m} \{ \sum_{l=1}^{k} (F(l) + p_l')/k \}$.

Theorem 4 shows that OSL computed in Step 3 of Algorithm B is a lower bound of the schedule length of an optimal schedule. We shall show in Theorem 5 that Step 4 of Algorithm B can always schedule the rank r tasks into the block (PF, OSL), where PF is the processor profile of the $SPT^{\#}$ schedule constructed in Step 1 of Algorithm B. Note that the rank r tasks are scheduled in nonincreasing order of their execution times. Task T_{n-l+1} is scheduled at the l^{th} iteration of Step 4 of Algorithm B, $1 \leq l \leq m$. The next three lemmas are instrumental in proving Theorem 5.

Lemma 2 : In Step 4 of Algorithm B, the number of available processors remained after the l^{th} iteration is $m - l$, $1 \leq l \leq m$.

Lemma 3 : Let $F(k, l)$, $1 \leq k \leq m$, $1 \leq l \leq m$, denote the finishing time of processor k before the l^{th} iteration in Step 4 of Algorithm B. Let processors $k_1 < k_2 < \ldots < k_{m-l+1}$ be the available processors before the l^{th} iteration Then, we have $F(k_1, l) \leq F(k_2, l) \leq \ldots \leq F(k_{m-l+1}, l)$.

Lemma 4 : Let processors $k_1 < k_2 < \ldots < k_{m-l+1}$ be the available processors before the l^{th} iteration in Step 4 of Algorithm B. If task T_{n-l+1} is scheduled by rules (1) or (2) in the l^{th} iteration on processor k_c, or if T_{n-l+1} is scheduled by rule (3) on processors k_c and k_{c+1}, then there are at least $c - 1$ available processors after T_{n-l+1} is scheduled that have processing capacity at least p_{n-l+1}.

Theorem 5 : Step 4 of Algorithm B can always schedule the tasks T_n, T_{n-1}, ... , T_{n-m+1} into the block (PF, OSL), where PF is the processor profile of the $SPT^{\#}$ schedule constructed in Step 1 and OSL is computed in Step 3 of Algorithm B.

From the above theorems, we have the main result of this paper which is stated in the following theorem.

Theorem 6 : Algorithm B always generates an optimal schedule.

4. CONCLUDING REMARKS.

In this paper, we have given an O(n log n) time algorithm to find a minimum length preemptive schedule subject to the constraint that it must have the minimum flow time. As a generalization of our problem, it will be interesting to find a minimum length preemptive schedule subject to the constraint that the flow time is not larger than a given parameter, α. One can also generalize this problem to the case of uniform processors. We note that, for the case of uniform processors, Gonzalez [8] has given an O(nm) time algorithm to find a minimum flow time preemptive schedule subject to the constraint that the schedule length is not larger than a given parameter, β.

REFERENCES

1. Bruno, J. L., E. G. Coffman, Jr., and R. Sethi, "Algorithms for Minimizing Mean Flow Time," *Information Processing 74*, Amsterdam : 1974, pp. 504-510.
2. Coffman, E. G., Jr., M. R. Garey, and D. S. Johnson, "An Application of Bin-Packing to Multiprocessor Scheduling," *SIAM J. on Computing,* Vol. 17, 1978, pp. 1-17.
3. Coffman, E. G., Jr., M. R. Garey, and D. S. Johnson, "Bin Packing with Divisible Item Sizes," *Journal of Complexity,* To appear.
4. Coffman, E. G., Jr. and R. Sethi, "Algorithm Minimizing Mean Flow Time : Schedule-Length Properties," *Acta Informatica,* Vol. 6, 1976, pp. 1-14.
5. Conway, R. W., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling,* Addison-Wesley, Reading, Mass., 1967.
6. Garey, M. R. and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-completeness,* Freeman, San Francisco, 1979.
7. Gonzalez, T., "Minimizing the Mean and Maximum Finishing Time on Identical Processors," Technical Report CS-78-15, Computer Science Dept., The Pennsylvania State University, University Park, PA., 1978.
8. Gonzalez, T., "Minimizing the Mean and Maximum Finishing Time on Unifrom Processors," Technical Report CS-78-22, Computer Science Dept., The Pennsylvania State University, University Park, PA., 1978.
9. Graham, R. L., "Bounds on Multiprocessing Timing Anomalies," *SIAM J. on Applied Math,* Vol 17, 1969, pp. 416-429.
10. Hochbaum, D. S. and D. B. Shmoys, "Using Dual Approximation Algorithms for Scheduling Problems : Theoretical and Practical Results," *J. ACM,* Vol. 34, 1987, pp. 144-162.
11. Hong, K. S. and J. Y-T. Leung, unpublished results.
12. Langston, M. A., *Processor Scheduling with Improved Heuristic Algorithms,* Ph. D. Dissertation, Texas A&M University, College Station, TX, 1981.
13. Leung, J. Y-T., "On Scheduling Independent Tasks with Restricted Execution Times," *Operations Research,* Vol. 30, 1982, pp. 163-171.
14. McNaughton, R., "Scheduling with Deadlines and Loss Functions," *Management Science,* Vol. 12, 1959.
15. Sahni, S., "Algorithms for Scheduling Independent Tasks," *J. ACM,* 1976, pp. 116-127.
16. Sahni, S., "Preemptive Scheduling with Due Dates," *Operations Research,* Vol. 27, 1979, pp. 925-934.

T_i	T_1	T_2	T_3	T_4	T_5	T_6
P_i	2	5	6	6	8	10

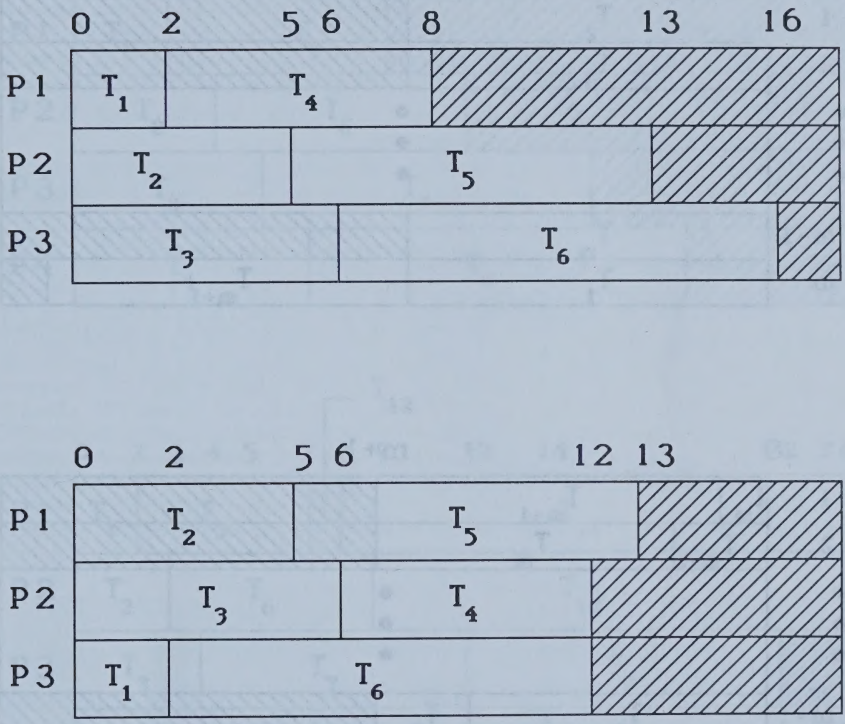

0
2
5 6
8
13
16
P1
P2
P3
T1
T4
T2
T5
T3
T6
0
2
5 6
12 13
P1
P2
P3
T2
T5
T3
T4
T1
T6

Figure 1. Two SPT Schedules for a set of Tasks.

T_i	T_1	T_2	T_3	• • •	T_m	T_{m+1}
P_i	m	m	m	• • •	m	m

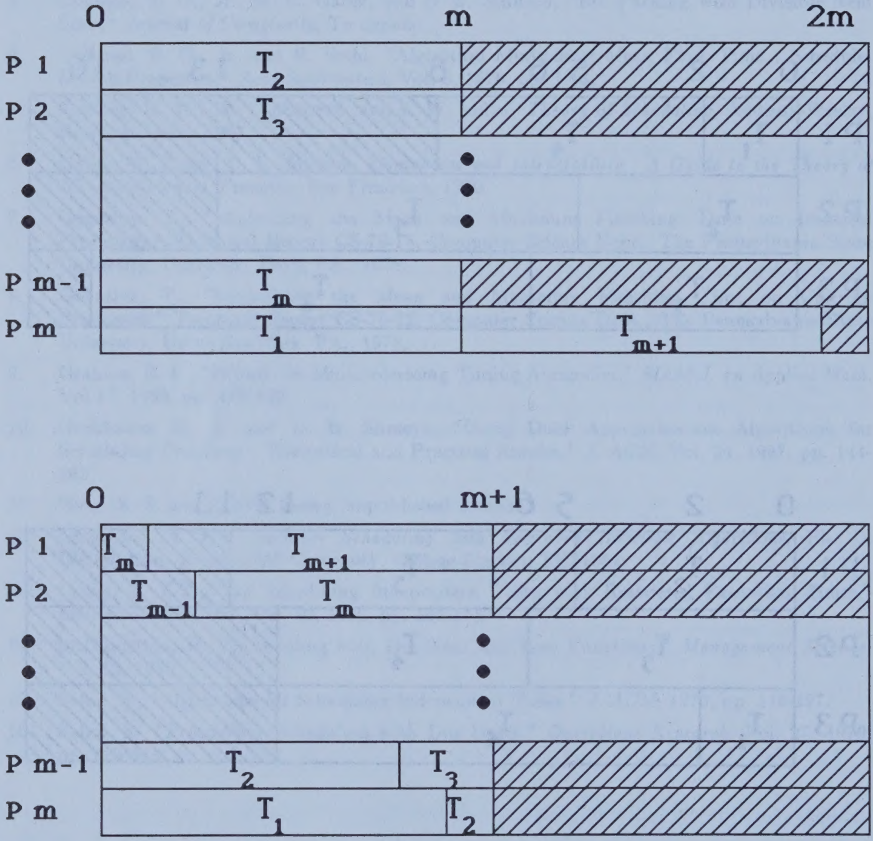

0
m
2m
P 1
T_2
P 2
T_3
P m-1
T_m
P m
T_1
T_{m+1}
0
m+1
P 1
T_m
T_{m+1}
P 2
T_{m-1}
T_m
P m-1
T_2
T_3
P m
T_1
T_2

Figure 2. A Task Set such that $SL(S^*_{NP})/SL(S^*_{P})=2m/(m+1)$.

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
P_i	2	3	4	5	5	6	8	9	10	10	16	16

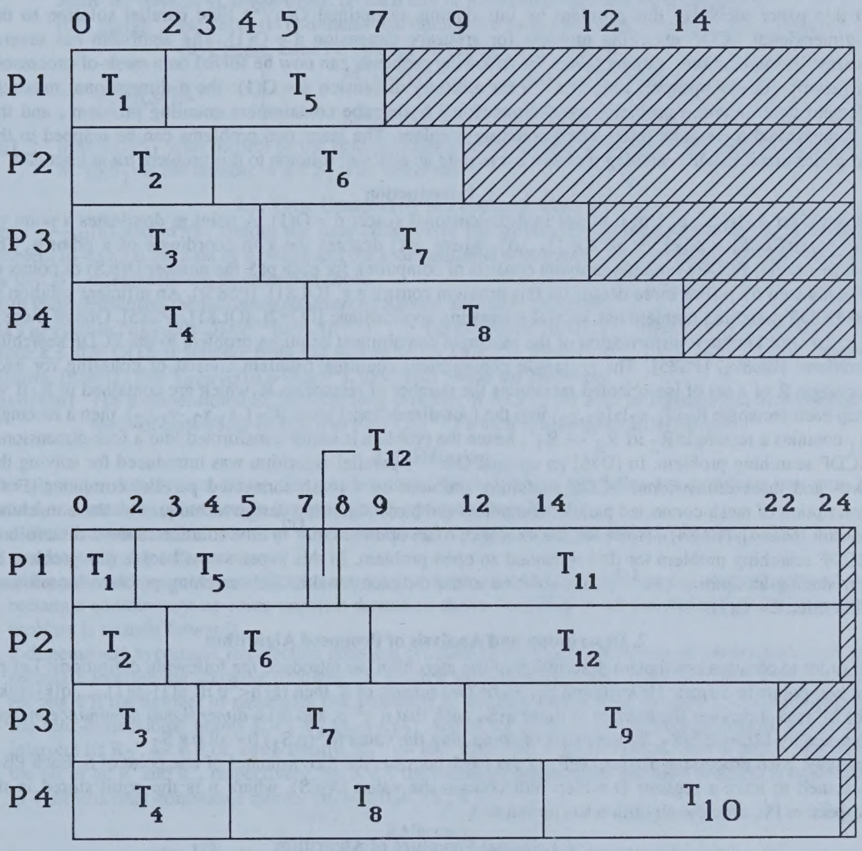


Figure 3. An Example Showing Algorithm B.

AN OPTIMAL PARALLEL SOLUTION TO THE ECDF SEARCHING PROBLEM FOR HIGHER DIMENSIONS ON A MESH-OF-PROCESSORS

FRANK DEHNE[1] *, IVAN STOJMENOVIC[2]
[1]School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
[2] Institute of Mathematics, University of Novi Sad, 2100 Novi Sad, Yugoslavia

Abstract

[D86] presented an optimal $O(n^{1/2})$ time parallel algorithm for solving the ECDF searching problem on a mesh-of-processors for a set of n points in two- and three-dimensional space. However, it remained an open problem whether such an optimal solution exists for the d-dimensional ECDF searching problem for $d \geq 4$.
In this paper we solve this problem by introducing an optimal $O(n^{1/2})$ time parallel solution to the d-dimensional ECDF searching problem for arbitrary dimension $d = O(1)$. The algorithm has several interesting implications. Among others the following problems can now be solved on a mesh-of-processors in (asymptotically optimal) time $O(n^{1/2})$ for arbitrary dimension $d = O(1)$: the d-dimensional maximal element determination problem, the d-dimensional hypercube containment counting problem , and the d-dimensional hypercube intersection counting problem. The latter two problems can be mapped to the 2d-dimensional ECDF searching problem but require an efficient solution to this problem for at least $d \geq 4$.

1. Introduction

Given a set $S=\{p_1,...,p_n\}$ of n points in d-dimensional space; $d = O(1)$. A point p_i dominates a point p_j ($p_i > p_j$), iff $p_i[k] > p_j[k]$ for all $k \ \varepsilon \ \{1,...,d\}$, where p[k] denotes the k-th coordinate of a point p. The d-dimensional ECDF searching problem consists of computing for each $p \varepsilon S$ the number D(p,S) of points of S dominated by p (For more details on this problem consult e.g. [OL81], [PS85]). An efficient solution to the ECDF searching problem has several interesting applications; [EO82], [OL81], [PS85]. One of these is e.g. the well known transformation of the rectangle containment counting problem to the ECDF searching problem; [EO82], [PS85]. The rectangle containment counting problem consist of counting for each rectangle R of a set of iso-oriented rectangles the number of rectangles R' which are contained in R. If we map each rectangle $R=[x_1,x_2]x[y_1,y_2]$ into the four-dimensional point $R'= (-x_1,x_2,-y_1,y_2)$ then a rectangle R_1 contains a rectangle R_2 iff $R_2' <= R_1'$, hence the problem is easily transformed into a four-dimensional ECDF searching problem. In [D86] an optimal $O(n^{1/2})$ parallel algorithm was introduced for solving the two- and three-dimensional ECDF searching problem on a mesh-connected parallel computer (For a description of mesh-connected parallel computers and basic algorithm design techniques on these machines consult [MS84], [UL84]). However, the existance of an optimal $O(n^{1/2})$ time solution to the d-dimensional ECDF searching problem for $d \geq 4$ remained an open problem. In this paper we will solve this problem by introducing an optimal $O(n^{1/2})$ time solution to the d-dimensional ECDF searching problem for arbitrary dimension $d = O(1)$.

2. Description and Analysis of Proposed Algorithm

In order to obtain a conveniant description of the algorithm we introduce the following definitions: Let p,q be two points in d-space ($1 \leq k \leq d$) and S_1, S_2 be two subsets of S, then (a) $q <^k p$ iff $q[1]<p[1], ..., q[k]<p[k]$, (b) $M^k(p,S_1)$ denotes the number of those $q \varepsilon S_1$ such that $q <^k p$, and (c) *k-dimensional dominance merge*, denoted by $MERGE^k(S_2, S_1)$, consists of computing the value $M^k(p,S_1)$ for all $p \ \varepsilon \ S_2$.

Initially, each processing element (PE) of the mesh contains the d coordinates of one point of S. Each PE is assumed to have a register D which will contain the value D(p,S), where p is the point stored in the respective PE, after the algorithm has terminated.

2.1. Global Structure of Algorithm

The global structure of the proposed algorithm is a divide-and-conquer mechanism which solves the problem as follows :

(I) Divide: Partition S into two subsets S_1 and S_2 by comparing the d-th coordinate of the points with their median d-th coordinate. S_1 and S_2 are stored in one half of the mesh-of- processors, each.
(This step is easily obtained by sorting S with resp. to the d-coordinate; see e.g. [TK77]. Split mesh by either a vertical or a horizontal line to minimize the diameter of the submeshes.)

(II) Recur: Solve the d-dimensional ECDF searching problem for S_1 and S_2, respectively, on each half of the mesh-of-processors in parallel.

(III) Merge (a) Solve (d-1)-dimensional dominance merge problem $MERGE^{d-1}(S_2, S_1)$.
(b) Update: Each PE updates his register D as follows:

$$D(p,S) := \begin{cases} D(p,S_1) & \text{for } p \ \varepsilon \ S_1 \\ D(p,S_2) + M^{d-1}(p,S_1) & \text{for } p \ \varepsilon \ S_2 \end{cases}$$

The following section shows how to solve the k-dimensional dominance merge problem $MERGE^k(S_2, S_1)$, $1 \leq k \leq d$, as required for step (III,a).

---

*Research supported by NSERC Grant No. A9173 .

2.2. K-Dimensional Dominance Merge $\mathrm{MERGE}^k(S_2, S_1)$

The structure of the k-dimensional dominance merge algorithm is again a divide-and-conquer mechanism. In each iteration k is decremented by one , i.e. the merge step for k-dimensional dominance merge involves the solution of a (k-1)-dimensional dominance merge problem. This process is iterated until k=1.

Each PE is assumed to have a register M which will finally contain the value $M^k(p,S_1)$ for all $p \varepsilon S_2$, where p is the point stored in the respective PE.

$k \geq 2$: (I) Divide: Partition S_1 into two subsets S_{11} and S_{12} and S_2 into two subsets S_{21} and S_{22} by comparing the k-th coordinate of the points with the median k-th coordinate of $S_2 \cup S_1$.

(II) Recur: Solve the k-dimensional dominance merge problems $\mathrm{MERGE}^k(S_{21},S_{11})$ and $\mathrm{MERGE}^k(S_{22},S_{12})$, respectively, on each half of the mesh-of-processors in parallel.

(III) Merge (a) Solve (k-1)-dimensional dominance merge problem $\mathrm{MERGE}^{k-1}(S_{22}, S_{11})$.

(b) Update: Each PE updates his register M as follows:

$$M^k(p,S_1) := \begin{cases} M^k(p,S_{11}) & \text{for } p \varepsilon S_{21} \\ M^k(p,S_{12}) + M^{k-1}(p,S_{11}) & \text{for } p \varepsilon S_{22} \end{cases}$$

$k=1$: Sort $S_2 \cup S_1$ with respect to the first coordinate in snake like ordering [TK77]. For each $p \varepsilon S_2$, $M^k(p,S_1)$ is the number of $q \varepsilon S_1$ with lower rank.

2.3. Time Complexity of Proposed Algorithm

Let $T_{ECDF}(n)$ and $m^k(n)$ denote the time complexity for solving the d-dimensional ECDF searching problem ($d = O(1)$) for a set of n points and the k-dimensional dominance merge problem $\mathrm{MERGE}^k(S_2,S_1)$ for $|S_2 \cup S_1|=n$, respectively, as described above.

With these definitions the following recurrence relations are easily observed:

(1) $T_{ECDF}(n)=T_{ECDF}(n/2)+m^{d-1}(n)+O(n^{1/2})$ (2) $m^k(n)=m^k(n/2)+m^{k-1}(n)+O(n^{1/2})$, $m^1(n)=O(n^{1/2})$.

Since $k \leq d = O(1)$, it follows from (2) that $m^k(n) = O(n^{1/2})$. Hence, $m^{d-1}(n) = O(n^{1/2})$ and, thus, it follows from (1) that $T_{ECDF}(n) = O(n^{1/2})$, too. This yields the following

theorem: The d-dimensional ECDF searching problem, $d = O(1)$, for a set of n points can be solved on a mesh-of-processors of size n in time $O(n^{1/2})$ which is asymptotically optimal.

3. Conclusion

The algorithm has several interesting implications. Among others the following problems can now be solved on a mesh-of-processors in (asymptotically optimal) time $O(n^{1/2})$:

- d-dimensional maximal element determination ($d = O(1)$): compute the set of points which are not dominated by any other point
- d-dimensional hypercube containment counting problem ($d = O(1)$): d-dimensional generalization of the rectangle containment counting problem described above (mapping to 2d-dimensional ECDF searching problem is straight forward)
- d-dimensional hypercube intersection counting problem ($d = O(1)$): d-dimensional generalization of the rectangle intersection counting problem; i.e. given a set S of iso-oriented rectangles determine for each rectangle R the number of rectangles that intersects R. Each rectangle $R=[x_1,x_2]x[y_1,y_2]$ is mapped into the four-dimensional points $R'= (-x_1,x_2,-y_1,y_2)$ and $R''= (-x_2,x_1,-y_2,y_1)$. Two rectangles R_1 and R_2 intersect iff $R_2'' <= R_1'$ or, equivalently, $R_1'' <= R_2'$; [EO82], [PS85]. Hence, with S' and S" denoting the set of all R' and R", respectively, for $R \varepsilon S$ the rectangle intersection counting problem is equivalent to 4-dimensional dominance merge, i.e. $\mathrm{MERGE}^4(S', S'')$.

References

[D86] F.Dehne, "$O(n^{1/2})$ Algorithms for the Maximal Elements and ECDF Searching Problem on a Mesh-Connected Parallel Computer", Information Processing Letters, 22, 1986, pp.303-306

[EO82] H.Edelsbrunner, M.H.Overmars, "On the Equivalence of Some Rectangle Problems", Information Processing Letters, 14, 3, 1982, pp.124-127

[H85] S.Hambrusch, private communication

[MS84] R.Miller, Q.F.Stout, "Computational Geometry on a Mesh-Connected Computer", Proc. IEEE 1984 Int. Conf. on Parallel Processing, pp.66-73

[OL81] M.H.Overmars, J.v.Leeuwen, "Maintenance of Configurations in the Plane", Report RUU-CS-81-3, Dept. of Computer Science, Univ. of Utrecht, Febr. 1981

[PS85] F.P.Preparata, M.I.Shamos, "Computational Geometry, An Introduction", Springer-Verlag, New York, 1985

[TK77] C.D.Thompson, H.T.Kung, "Sorting on a Mesh-Connected Parallel Computer", C.ACM, Vol.20, No.4, April 1977

[UL84] J.D.Ullman, "Computational Aspects of VLSI", Principles of Computer Science Series, Computer Science Press, 1984

SPORT-SORT

Sorting Algorithms and Sport Tournaments

HANS BLOCK
Department of Computer and Information Science, Linköping University, S-581 83 LINKÖPING, Sweden, and Statistics Sweden, S-115 81 STOCKHOLM, Sweden

1 THE PROBLEM

We want to sort n elements under certain conditions, using $n / 2$ processors for the comparisons. All other computations are considered to be very fast and can be executed by one single processor. We want to minimize the number of **rounds**, or sets of simultaneous comparisons. The time and memory needed for the other calculations should not be too great.

The problem is motivated by sport tournaments. These can be regarded as sorting problems in which the comparison time is much greater than the computation time. Each team can be compared to at most one other team at a time. We assume that the relation "A is better than B" is transitive, but we cannot be sure, since the outcome of real matches have a stochastic element. Matches which might give loops in this relation are forbidden. The goal is to give the complete ranking list in as few rounds as possible.

The application can be formulated in the language of graph theory: Given a directed graph with n nodes. In the beginning there are no arcs. In each round, add several arcs at the same time, at most one for each node, in such way that no loops occur, whatever directions the new arcs are given. After the round, assign an arbitrary direction to every new arc. Continue until the graph is a totally ordered set. The goal is to minimize the average number of rounds.

2 AN ALGORITHM AND ITS PROPERTIES

We have constructed several algorithms which solves this problem. The best of them will be described here. First we define the points of team i at a certain stage of the tounrnament as

points(i) = # of teams better than i -
- # of teams worse than i

"Teams better than i" are teams which have once defeated i, directly or indirectly, using the transitivity.

The algorithm runs as follows:

```
Mix the teams in random order;
REPEAT
        Allocate matches between adjacent teams when possible;
        Play the matches;
        Draw all possible conclusions;
        Sort the teams according to their points;
        Mix teams with equal points randomly;
UNTIL   all teams have all relations fixed.
```

This algorithm solves our problem, according to the following two theorems:

Theorem 1. If the teams are sorted by their points as defined above, they are also topologically sorted with respect to the relation "A has directly defeated B".

Theorem 2. If the matches are played between adjacent teams on a topologically sorted ranking list, every team participates in at most one match at a time, and there is no direct relation between teams in a match, then no loops can occur after the round.

The proofs are almost immediate from the definitions. Our main result is

Theorem 3. The algorithm takes logarithmic time, with a small constant.

Outline of proof. We assume that the result of a match is always determined by the inherent, **hidden strengths** of the participating teams. We also define

rank(i, k)	Place in ranking list of team with hidden strength *i* after round *k*.
D_k	$\sum_i (rank(i, k) - i)^2$, which we call the **disorder** after round *k*.

For random permutations, $D_0 = n^3 / 6$. We will hint that the disorder is approximately halved in most rounds of the tournament. Since the tournament is finished when the disorder is *1*, the number of rounds will be $3 \cdot lg\ n$.

rank(i, k) can be regarded as a stochastic variable. Experience and arguments of symmetry tell that the distribution of *rank(i, k) - i* has mean *0* and is the same for most *i*, but depends on *k* as a parameter.

If team *i* has won over team *j* in round k_0, this implies that

$$rank(i, k) < rank(j, k) \quad for\ all\ k >= k_0$$

for the points defined above.This means that the distribution of *rank(i, k)* is cut off at the point *rank(j, k)*, which is also a stochastic variable with mean *i*. For all *k*, the cut distribution will thus have a variation which is a constant times the original variation. The variation of the distribution, and also the disorder, will be reduced by the same constant in each round.

Assumptions of normality gave the value *0.519...* to this constant, predicting $3.17 \cdot lg\ n - 2.65$ rounds. This is in good agreement with experiments which at the average took $2.60 \cdot lg\ n - 2.56$ rounds.

3 IMPLEMENTING THE ALGORITHM

We store the victories and defeats of each team. We also need a few more arrays of *n* words. Since each round gives at most one victory or defeat for each team, $n \cdot (c + 3 \cdot lg\ n)$ words of memory are needed, due to theorem 3. During the searching, unnecessary results can be removed, reducing the memory requirements.

Calculating the points is the most time-consuming job. The teams better than *i* are found by starting from team *i*, following the arcs of defeats until we meet a team with no defeats or until we come to a team a second time. The losers to *i* are calculated by similar, recursive searching in the network.

We reach each node at most once and make at most *r* tests per node, where *r* is the maximal number of rounds. Since the searching is done for each team in every round, there will be at most $n^2 \cdot r^2$ tests. Since $r < 3 \cdot lg\ n$, by experiences and theorem 3, the time will be $O(n^2 \cdot lg^2\ n)$.

The searching can in fact be stopped earlier. We need not investigate relations which have been established in previous rounds. Arguments similar to those in theorem 3 hint that the depth of investigation decreases geometrically by the number of rounds, so the whole computing time will be only $O(n^2 \cdot lg\ n)$.

Summing up: Even with the restrictions of our problem, we have come very close to the theoretical bound for the number of rounds in parallel sorting. The computing time and memory requirements are quite modest, so the algorithm can be implemented on personal computers for all real tournaments.

The participants in real tournaments will find our scheme interesting, since they will meet teams more and more equal in strength, as the competition goes on. All teams are guaranteed to play *lg n* matches, and most of them will play in most rounds. Very large tournaments can be arranged without previous seeding. Competitions with several thousands of teams could be settled in *20 - 30* rounds.

Reference. Hans Block: Sport-Sort, Linköping Studies in Science and Technology, Thesis No 111, Linköping., Sweden 1987.

MODELING AND DYNAMIC SCHEDULING OF A QUEUEING SYSTEM WITH BLOCKING AND STARVATION

Rodolfo A. Milito and Hanoch Levy
AT&T Bell Laboratories, Holmdel, NJ 07733

ABSTRACT

We consider the problem of dynamically controlling N queues that compete for the service attention of a single server. The system under study consists of *finite* queues and the performance objective is to avoid situations in which any of the queues are *full* (*blocking*). We also consider a system in which the queues are *fed* by the server and the objective is to avoid situations in which any of the queues are ***empty*** (***starvation***), and a ***hybrid system***, consisting of both types of queues. The goal of this paper is to characterize the optimal decision rule for controlling these systems. From the properties derived in the paper we conclude that it is unlikely that a simple form optimal rule can be found. However, we are able to use these properties to construct a heuristic decision rule which resembles the optimal rule and which is numerically shown to closely approximate it.

1. Introduction

We consider the problem of dynamically controlling N queues that compete for the service attention of a single server. In contrast to previous papers which studied similar systems where the queues are *infinite* and the performance objective is linear in the *number of customers in the system,* we concentrate in this work on studying a system where the queues are *finite* and the performance objective is to minimize the probability of the queues being *full.*

It is common to consider delay (and especially its first moment) as the main performance criterion of a queueing system and for this reason the previous studies considered a performance objective which is linear with the queue occupancy. However, in many queueing systems a not less important measure is the *blocking* of customers – the system incurs a penalty if a customer finds the system full and cannot join it. For other systems, such as those in which we are interested in this paper, blocking is the main performance objective, while delay is irrelevant.

As examples let us describe the systems that are covered by this study. First, consider a computer which is connected, in a star-topology, to N work stations. System users continuously input jobs at the work stations; after being input, each job is shipped to the central computer for further processing and storage. Once a user is done with inputting a job into the local station he/she is not concerned any more with that job and is ready to process the next one. The job itself is still stored in the local station until the computer polls the station and retrieves the job. The station is equipped with an output buffer (queue) which is used to temporarily store jobs which have been input by the user but which have not been retrieved yet by the computer. However, since these buffers are finite, the user may reach a situation where the station's buffer is *full.* At that moment the station is *blocked* and the user is temporarily prevented from working. Such a system is penalized for the periods in which there is blocking. That is, the main performance criterion is the blocking of jobs in the local station and *not the amount of time a job stays in the queue.* The problem of the central computer in such a system is which station to serve so as to optimize the system performance according to that criterion. We call this system a *blocking system.*

A similar system, which we call a *starvation system,* is one consisting of a central computer and N stations, connected in the same configuration described above. However, in this system the computer ships jobs to the peripheral stations, where they are then processed by users. Here the local station is equipped with an input buffer to store incoming jobs, and the main goal is to keep the stations busy, or in other words, to prevent them from *starvation*. Starvation is the situation where the input buffer of the station is empty and the user must stop working until a new job is shipped from the central computer. Systems which contain both type of stations, i.e., blocking stations and starvation stations, are called *hybrid systems.*

In this paper we consider the problem of optimally controlling these types of systems. For simplicity of the mathematical exposition we will consider in our analysis only the starvation system. The extensions of the results to blocking systems and hybrid systems are trivial. We assume that the time to feed a job from the central computer to the local station and the time to process this job by the dedicated server of the local station (the user) are both exponentially distributed. The cost associated with queue i being in starvation at time t is assumed to be C_i and the objective function is the sum of the future discounted costs of all queues. The goal in the paper is to characterize the optimal control policy of this system.

We study the system by investigating the properties of the optimal decision rule. We concentrate on examining properties which are usually possessed by simple form rules, and obtain the following results: 1) We prove that the optimal decision rule for a two-queue system with identical feeding rates and identical service rates is characterized by a switching curve. 2) We show that the switching curve is monotonically increasing, and derive its characteristics as function of the cost parameters. 3) We observe that the optimal decision rule is quite insensitive to the queue size; However, we prove that the rule is not completely independent of the queue size. 4) We prove that the optimal decision rule for an N queue system is not always decomposable.

Having realized that the optimal decision rule does not *always* possess all the desired properties, we conjecture that a simple closed-form optimal decision rule is unlikely to be found. Therefore, we concentrate on proposing a heuristic rule which will approximate the optimal decision rule. The selection of this rule is done by matching its properties to those of the optimal rule. A numerical comparison shows that the heuristic rule performs very closely to the optimal rule under wide variety of conditions.

The structure of this paper is as follows: In Section 2 we give a detailed formal description of the model. In Section 3 we analyze the properties of a two-queue system. In Section 4 we discuss a system consisting of N queues. In Section 5 we propose a heuristic rule and compare its performance with that of the optimal rule. Due to lack of space, we have eliminated, throughout the paper, the detailed proofs of the theorems stated. The reader may find them in [1].

1.1 *Previous work*

Space limitations prevent us from placing this paper in the context of *normative* queueing systems. The interested reader may consult Milito and Levy[1] where the following key papers are referenced: Baras, Dorsey and Makowski [2], Buyukkoc, Varaiya and Walrand [3], Gittins[4], Klimov[5], [6], Stidham[7], Varaiya, Walrand and Buyukkoc[8] and Whittle[9].

2. Mathematical formulation of the problem

In this section we formulate the *starving* problem described in the introduction in mathematical terms. Our main concern is to keep the buffers non empty to avoid *starvation*. The system consists of one central server and N queues. Whenever Q_i is not empty, customers depart from it at rate λ_i. (In physical terms, it means that Q_i is served by a dedicated exponential server of rate λ_i). The task of the central server is to *feed* the queues with customers. The time it takes the central server to *feed* Q_i with a customer is exponentially distributed with parameter μ_i. This central server knows the parameters of the system as well as the state of the system (number of customers in the different queues) at each instant of time. The goal is to use this information in the best possible way, i.e., to find a *feeding* policy that minimizes the following discounted (over time) weighted (over the different queues) sum of the probabilities of the queues being empty:

$$J_\Pi^\alpha\left(x,\ T\right) = E_\Pi\left\{\int_0^T\left[\sum_{i=1}^{N} C_i\,\mathbf{1}(\mathbf{x}_i(t)=0)\right]e^{-\alpha t}dt\ \middle|\ \mathbf{x}(0)=x\right\}, \tag{2.1}$$

where:
$\mathbf{x}_i(t) =$ *number of customers in* Q_i *at time* t,

$\mathbf{x}(t) = (\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t))$,

$\alpha > 0$, $\mathbf{1}(\cdot)$ stands for the indicator function, and Π denotes a policy which assigns the central server to one of the queues at each instant of time. The so called *discount rate* α penalizes present *starvation* more heavily than future *starvation.*[1]

Our problem is to find an optimal policy Π^*, which minimizes this cost functional within the class of non-anticipative, preemptive (either *preemptive resume or preemptive repeat with resampling*[2]), perfect state information policies. In our context, perfect state information means that the controller knows at each instant the exact number of customers in the different queues.

So far, we have considered the problem in its natural continuous time framework. We seek to convert our problem into a Markovian Decision Process, for which a well developed body of theory exists. In order to recast the problem into a discrete time framework we follow Lippman's [10] "uniformization" approach. (See Serfozo [11] for an elaboration of Lippman's approach and Hajek [12] for an application of the method to a quite general problem of dynamic scheduling of two queues).

Let the *total transition rate* in our system be

$$r = \sum_{i=1}^{N} [\lambda_i + \mu_i] \tag{2.2}$$

This transition rate assumes that the central server *feeds* all queues simultaneously, and the dedicated servers are always active. The probability of having a *potential arrival (departure)* to (from) Q_i is μ_i / r (λ_i / r). A *potential arrival* to Q_i becomes a real arrival iff the central server is actually feeding Q_i. A *potential departure* from Q_i becomes real if $x_i > 0$. *Potential transitions* that do not materialize into real ones do not affect the state of the system. It can easily be shown that the cost to go for an horizon of n transitions, when the initial state is x, is given by $\dfrac{1}{\alpha + r} V_n^\beta(x)$, where

$$V_n^\beta(x) \triangleq \min_{\Pi} E_\Pi \left\{ \sum_{k=0}^{n-1} \beta^k \sum_{i=1}^{N} C_i \, \mathbf{1}(\mathbf{x}_i(k) = 0) \mid \mathbf{x}(0) = x \right\}, \tag{2.3}$$

the factor $\beta \triangleq \dfrac{r}{\alpha + r}$ is the discrete time counterpart of the discount rate α, and the index k denotes the k^{th} transition epoch. It follows that $\lim_{T \to \infty} \min_{\Pi} J_\Pi^\alpha(x, T) = \dfrac{1}{\alpha + r} \lim_{n \to \infty} V_n^\beta(x)$. In order to ease the notation, from now on we will drop the superscript in V_n^β, except when an explicit reference to the discount factor is needed. Also, without loss of generality, we will normalize the parameters of the system, so that $r = 1$.

Recall the definition of the state vector $x = (x_1, x_2, \ldots, x_N)$, and define the arrival and departure operators as follows:

$$\mathbf{A}_i x = \begin{cases} (x_1, \ldots, x_i + 1, \ldots, x_N) & \text{if } x_i < L_i \\ x & \text{otherwise} \end{cases} \tag{2.4}$$

$$\mathbf{D}_i x = \begin{cases} (x_1, \ldots, x_i - 1, \ldots, x_N) & \text{if } x_i > 0 \\ x & \text{otherwise} \end{cases} \tag{2.5}$$

The cost to go V_n is then characterized by the Dynamic Programming equation:

---

[1] The introduction of a discount facilitates the analysis because of the contraction property possessed by the optimal cost to go. From a practical perspective, the discount accounts for uncertainties on the planning horizon [9].

[2] Since the service time distribution is exponential, *preemptive resume* and *preemptive repeat with resampling* are equivalent.

$$V_0(x) = 0 \tag{2.6a}$$

$$V_{n+1}(x) = \mathbf{T}V_n(x) \tag{2.6b}$$

where the operator **T** is defined by

$$\mathbf{T}f(x) \triangleq \sum_{i=1}^{N} C_i \, \mathbf{1}(x_i = 0) + \beta \sum_{i=1}^{N} \lambda_i \, f(\mathbf{D}_i x) + \beta \frac{\mu_{i^*}}{r} f(\mathbf{A}_{i^*} x) + \beta \sum_{i \neq i^*} \frac{\mu_i}{r} f(x) \tag{2.7}$$

and

$$i^* \triangleq arg \min_i \left\{ \frac{\mu_i}{r} f(\mathbf{A}_i x) + \sum_{j \neq i} \frac{\mu_j}{r} f(x) \right\}.$$

Since the cost is bounded, and $\beta < 1$, it is easy to see that **T** is a contraction. Let $V(x) = \lim_{n \to \infty} V_n(x)$ denote the optimal cost to go for the infinite horizon problem. Then, $V(x)$ is the solution of the fixed point equation, in this context commonly referred to as "Bellman's equation" or "Optimality Equation".

$$V(x) = \mathbf{T}V(x) \tag{2.8}$$

This yields the following optimal decision rule:

$$\mu_i \Big[V(x) - V(\mathbf{A}_i(x)) \Big] \underset{Q_i \ Q_j}{\overset{Q_i}{\gtrless}} \mu_j \Big[V(x) - V(\mathbf{A}_j(x)) \Big] \tag{2.9}$$

where the notation $\underset{Q_j}{\overset{Q_i}{\gtrless}}$ means that the optimal server prefers to serve Q_i (Q_j) when the inequality $>$ ($<$) prevails.

3. A System Consisting of Two Queues

We start our study by looking at the simplest configuration: a system consisting of two queues. Any stationary policy for this system can be represented by an $L_1 + 1$ by $L_2 + 1$ matrix; let D denote the matrix representing the optimal (stationary) policy. The value given to the entry $D(x_1, x_2)$ represents the decision taken by the server when the number of customers in the buffers are x_1 and x_2. The entry can either take the values 1 or 2 (serve queue 1 or 2, respectively) or the value 0 (be indifferent between serving queue 1 or queue 2). The expected cost to go of this system can be represented in a matrix $V(x_1, x_2)$ whose dimension is identical to the dimension of D, and whose (i, j)th entry contains the expected cost to go when the buffer occupancies are i and j. In this section we study the structure of the matrices D and V as representing this system and the optimal decision algorithm.

3.1 Basic Properties of the Value and Optimal-Decision Matrices

The optimal decision matrix and the value function for the two-queue problem possess two important properties: 1) (Strict) monotonicity of the value function, and 2) The existence of a monotonically increasing switching curve which partitions the state space into two regions: a region in which it is optimal to serve Q_1 and a region in which it is optimal to serve Q_2. We next prove these properties (the proof of the second property is limited to the case of identical rates).

3.1.1 Monotonicity of the Value Function

PROPOSITION 1: Let $0 \leq x_1 < L_1$ and $0 \leq x_2 \leq L_2$. If $C_1 > 0$, then $V(x_1, x_2) > V(x_1 + 1, x_2)$. Similarly, for $0 \leq x_1 \leq L_1$ and $0 \leq x_2 < L_2$, if $C_2 > 0$ then $V(x_1, x_2) > V(x_1, x_2 + 1)$.

The proposition may be easily proved by observing the behavior of two systems (one whose initial state is (x_1, x_2) and the other whose initial state is $(x_1 + 1, x_2)$) under the same sample path.

3.1.2 The Existence of A Monotonic Switching Curve. An intuitive property possessed by the optimal-decision matrix is that $D(x_1, x_2) = 1$ implies $D(x_1 - 1, x_2) = 1$ and $D(x_1, x_2 + 1) = 1$. The idea behind this property is that if the server prefers to serve queue 1 when the station occupancy is (x_1, x_2) it should take the same decision when the occupancy of buffer 1 is smaller (and the occupancy of buffer 2 is the same). Similarly, it should take the same decision when the occupancy of station 2 is larger (and the occupancy of station 1 is the same).

While this property sounds "intuitive", its proof is not straightforward. This property was proved by Hajek[12] to hold for a fairly complex system but under the assumption that the cost function is increasing (in the buffer occupancy) and linear. In contrast, the cost function considered here is decreasing and non-linear, and thus requires different treatment. For simplicity of the analysis we consider in the sequel a symmetric system, i.e., a system in which $\lambda_1=\lambda_2=\lambda$ and $\mu_1=\mu_2=\mu$. The claim for the non-symmetric system can be proved by the same approach but requires more detail. Also, we assume a system with infinite buffers (Note that infinite queues are meaningful only in the context of the starvation system). The extension for the finite buffer system can be proved by observing that the upper boundaries of the two dimensional space (namely, the lines representing $x_1=L_1$ and $x_2=L_2$) can be viewed as the lower boundaries (namely $x_1=0$ and $x_2=0$) in Hajek's model[12] and by applying his analysis to our system.

The property is proved in two major steps. First, (Lemma 2) we show that the difference between the values of two adjacent points on the grid is bounded by a simple function of the cost parameters (C_1 and C_2). Second, in Theorem 3 and Corollary 4 we state the monotonicity of the switching curve. The main difference between the system analyzed here and the system analyzed by Hajek[12] is that here in order to show the correctness of the inductive step in the main theorem (namely, Theorem 3) one needs first to establish the inequality stated in Lemma 2. In contrast, in the system analyzed by Hajek[12] the inductive step of the main theorem can be proved directly. Lemma 2 and Theorem 3 are stated next:

LEMMA 2: For every x, n the following inequalities hold:
a) $C_2 + \beta\lambda[V_n(\mathbf{A}_2 x) - V_n(x)] > 0$; b) $C_1 + \beta\lambda[V_n(\mathbf{A}_1 x) - V_n(x)] > 0$

THEOREM 3: The differences $V_n(\mathbf{A}_2 x) - V_n(\mathbf{A}_1 x)$ and $V(\mathbf{A}_2 x) - V(\mathbf{A}_1 x)$ are (a) increasing in x_2, and (b) decreasing in x_1.

Lemma 2 is proved by induction on n and Theorem 3 is proved by induction on n and by making use of Lemma 2. The detailed proofs may be found in [1]. An immediate result from Theorem 3 is the monotonicity of the switching curve as presented in the following corollary:
COROLLARY 4: If $D(x_1,x_2)=1$ or $D(x_1,x_2)=0$, then $D(x_1,x_2+1) = D(x_1-1,x_2) = 1$. Similarly, If $D(x_1,x_2)=2$ or $D(x_1,x_2)=0$, then $D(x_1,x_2-1) = D(x_1+1,x_2) = 2$.

3.2 Relations Between the Cost Function and the Shape of the Switching Curve

The shape (and location) of the switching curve depends on the system parameters. In particular, an intuitive observation is that the curve should move toward the ordinate (assuming that the ordinate is the line representing $x_1=0$ and the abscissa is the line representing $x_2=0$) when C_2 increases, and toward the abscissa when C_1 increases. In the following analysis we prove this property. For the simplicity of the analysis we concentrate on the case where $\mu_1=\mu_2$ (however, we allow $\lambda_1\neq\lambda_2$) while the case of nonequal μ_i's can be proved in a similar manner.

The approach we take is to separate the value function into two parts, one representing the expected discounted cost related to future "landing" on the abscissa ($x_2=0$) and the other is related to future "landing" on the ordinate ($x_1=0$). For a given policy Π, we define these costs as:

$$f_\Pi(x) \overset{\Delta}{=} E_\Pi\left\{\sum_{s=0}^{\infty} \beta^s \mathbf{1}(\mathbf{x}_1(s)=0 \mid \mathbf{x}(0)=x))\right\} \tag{3.1a}$$

$$g_\Pi(x) \overset{\Delta}{=} E_\Pi\left\{\sum_{s=0}^{\infty} \beta^s \mathbf{1}(\mathbf{x}_2(s)=0 \mid \mathbf{x}(0)=x)\right\} \tag{3.1b}$$

Using this formulation, one may write the expected cost to go as: $J_\Pi(x)= C_1 f_\Pi(x)+ C_2 g_\Pi(x)$ and we may conclude:

LEMMA 5: For any fixed policy Π (not necessarily optimal), the expected cost to go $J_\Pi(x)$ is linear in C_1 and C_2.

Let us now hold C_1 and C_2 constant and let Π^* be the optimal policy for these values. Then the optional policy at x is taken according to: $\mu_1[V(x) - V(\mathbf{A}_1 x)] \underset{Q_2}{\overset{Q_1}{\gtrless}} \mu_2[V(x) - V(\mathbf{A}_2 x)]$. If we assume $\mu_1 = \mu_2$, substitute (3.1) into this equation and rearrange it, then the decision taken is according to

$$C_1\left[f_{\Pi^*}(\mathbf{A}_2 x) - f_{\Pi^*}(\mathbf{A}_1 x)\right] \underset{Q_2}{\overset{Q_1}{\gtrless}} C_2\left[g_{\Pi^*}(\mathbf{A}_1 x) - g_{\Pi^*}(\mathbf{A}_2 x)\right]. \tag{3.2}$$

Let us define $h_1(x) \overset{\Delta}{=} f_{\Pi^*}(\mathbf{A}_2 x) - f_{\Pi^*}(\mathbf{A}_1 x)$ and $h_2(x) \overset{\Delta}{=} g_{\Pi^*}(\mathbf{A}_1 x) - g_{\Pi^*}(\mathbf{A}_2 x)$. It is now easy to see that the relation between the switching curve to C_1 and C_2 is directly implied from the property $h_i(x) \geq 0$.

The approach for proving the properties of Π^* is to hold this policy fixed and inductively to show that they hold for any horizon n. We, therefore, define $f_n^{\Pi}(x)$ as the expected discounted cost due to future "landing" on the ordinate, subject to horizon n and subject to a given (arbitrary) policy Π: $f_n^{\Pi}(x) \overset{\Delta}{=} E_{\Pi}\left\{\sum_{s=0}^{n-1} \beta^s \mathbf{1}(\mathbf{x}_1(s)=0 \mid \mathbf{x}(0)=x)\right\}$ which can be computed recursively (where the recursion index is n). Let us now assume that Π^*, the optimal policy, is followed, and drop any explicit reference to it.

Next, in Lemma 6 we establish some bounds on $f_n(\mathbf{A}_2 x) - f_n(x)$ and on $f_n(\mathbf{A}_1 x) - f_n(x)$, and in Lemma 7 we state that $h_i(x) \geq 0$.

LEMMA 6: $1 + \beta\lambda_1[f_n(\mathbf{A}_2 x) - f_n(x)] > 0$ and $1 + \beta\lambda_1[f_n(\mathbf{A}_1 x) - f_n(x)] > 0$.

LEMMA 7: For $i = 1, 2$, $h_i(x) \geq 0$.

The proof of Lemma 6 is by induction on n and the proof of Lemma 7 is by induction on n and by making use of Lemma 6. Having established the properties stated in Lemma 7, we are now ready to state the main theorem of this section:

THEOREM 8: The function $Z(x, C_1, C_2) \overset{\Delta}{=} C_1 h_1(x) - C_2 h_2(x)$ is continuous and piecewise linear in both C_1 and C_2, increasing for C_1 and decreasing for C_2.

This result follows from the continuity of the value function $V(x)$. A rigorous proof is given in [1].

Theorem 8 implies that if the optimal decision for given costs C_1, C_2 is to serve Q_1 at state x, then this decision will hold for any C'_1, C'_2 such that $C'_1 \geq C_1$ and $C'_2 \leq C_2$. A symmetric statement holds for the service of Q_2. It is, therefore, concluded that the switching curve will move away from the abscissa and toward the ordinate when C_1 decreases and when C_2 increases.

3.3 Sensitivity to Buffer Size

An examination of the two dimensional system with equal rates has shown that the decision rule is quite insensitive to the buffer size. This means that as long as $L_1 > i$ and $L_2 > j$ the (i,j)th decision is close to being independent of L_1 and L_2. As a matter of fact the only sensitivity we have been able to observe by numerical tests is so weak, that it was not clear if it is due to buffer size or to numerical problems. For this reason we address this issue next, in a theoretical framework, and show that the optimal rule is *not completely independent* of the buffer size.

THEOREM 9: For a two-queue system, the decision rule is not independent of the buffer size.

The theorem is proved by considering two systems: System A where $L_1 = L_2 = 1$ and System B where $L_1 > 1$ and $L_2 > 1$, and by assuming (for the contradiction) that all their decisions are identical. We then consider the parameter values under which System A, when being in state (0,0), is indifferent between serving Q_1 and service Q_2. Using the Dynamic Programming equation we show that under those parameter values System B cannot be indifferent at (0,0). Therefore, by the continuity of the value function we conclude that there are parameter values for which System A's decision must be different from System B's decision. A rigorous proof is given in [1].

4. A System Consisting of N Queues

The difficulties observed above in characterizing the two-queue system suggest that it is probably more difficult to characterize the N-queue system. Yet it would be desired to relate the rule for the N-queue system to the optimal rule for the two-queue system. Maybe the most natural and desired property, in this regard, is the property of decomposition. This property, if shown to hold for this system, would imply that in order to find which of the N queues should be served when the buffer occupancy is $(x_1, x_2, \cdots, x_N)$ one could consider the following approach: First, consider a system consisting of Q_1 and Q_2 and find which of the queues should be served (this queue is to be called the *winner* of this comparison) when the buffer occupancy is (x_1, x_2). Next, pick the winner queue (assume it is Q_1) and compare it to Q_3 by considering a system consisting only of Q_1 and Q_3. For this system find which of the queues is to be served when the buffer occupancy is (x_1, x_3). Pick the winner of this comparison to "compete" with Q_4 and so on. The final winner (after performing $N-1$ such comparisons) is the queue to be served. The attractivity of such a scheme is the tremendous saving in computation in comparison to the computation of the optimal rule in an N queue system.

In this section we consider this issue of decomposition. We show that unfortunately the optimal decision rule is not decomposable under general conditions. However, we still propose to use decomposition as an approximation for the optimal rule. We tested this approximation for 486 different systems and found that the decomposition rule approximates very closely the optimal rule. The maximum relative error (in approximating the cost to go) observed in these tests was $4 \cdot 10^{-5}$. The details of this test may be found in [1].

4.1 The Optimal Decision Rule is not Generally Decomposable

Here we show that the optimal rule does not lend itself to decomposition under general conditions. To demonstrate this property, we consider a simple case of a system consisting of three queues whose lengths are $L_1 = L_2 = L_3 = 1$ and whose service rates are identical $\mu_1 = \mu_2 = \mu_3 = \mu = 1$. The other system parameters are: $\lambda_1 = 1$, $\lambda_2 = 1/2$, $\lambda_3 = 2$, $C_1 = 1$, $C_2 = 58/65$, $C_3 = 71/61$ and $\beta = 1/2$. In this simple system the only decision that may be subject to full decomposition is the one taken at (0,0,0), since in all other points either no decision is taken or the decision is between two queues.

Assuming that decomposition holds, we now consider three systems: System I consisting of Q_1 and Q_2, System II consisting of Q_2 and Q_3, and System III consisting of Q_3 and Q_1. We now use the Dynamic Programming equation and compute the decision taken by each of the systems for the state (0,0). The decisions for these parameter values are: System I and System III are indifferent while System II prefers serving Q_3 (on Q_1). We may now slightly increase C_2 and decrease C_3 and obtain a new set of parameters for which the following holds: System I prefers serving Q_2 (on Q_1), System II prefers serving Q_3 (on Q_2) and System III prefers serving Q_1 (on Q_3). Obviously, under these conditions general decomposition does not hold. A more rigorous derivation of this result may be found in [1].

The result derived above implies that a general decomposable rule does not exist for finite queue systems. Moreover, based on this result we conjecture that any simple closed-form decision rule for the system is unlikely to be found, since most of the known simple decision rules (like the μc rule[3]) are of the "index type"[5],[9],[4] and as such, decomposable.

5. A Heuristic Index Rule

Due to the difficulties faced with analyzing even simple properties of this system we believe that it is very difficult to describe the optimal rule with a closed form expression. For this reason, and to be able to use the system efficiently, we dedicate our efforts to devise a heuristic near optimal rule.

Inequality (2.9) shows the strong coupling between the dynamics of the two queues. We say that the coupling is strong because the expected gain obtained by serving Q_i is given by $\mu_i\,[V(x) - V(A_i(x)]$, which depends on the state x, and not only on its i^{th} component. If the coupling were weak, in the sense that this expected gain depended only on the i^{th} component, the optimal decision would be of the *index type*. (See e.g. [9] for a discussion of Gittins' Index Rule for Multiarmed Bandit Problems).

The characteristics of the switching curve suggest that the coupling is indeed very strong in the neighborhood of the switching curve, but vanishes when the system is away from it. This heuristic argument prompts us to look at each queue in isolation, with a fully dedicated server. This is a fairly good model for Q_i when the system is operating well into the Ω_i region, where Ω_i is the set of states in which the optimal decision is to serve Q_i, for i=1,2. Stretching the limits of this approximation, we will later use it to derive a heuristic rule of the *index type*. Direct comparison with the computed optimal policy will then be used to validate the heuristics.

5.1 A look at each queue in isolation

Consider a generic queue of buffer size L, arrival rate of customers μ, departure rate of customers λ, and total transition rate r. Let C be the cost incurred during the stages in which the queue is empty, β the discount factor and $W(i)$ the expected cost to go when the number of customers in the system is i. The fixed point equation for $W(i)$ reads:

$$W(i) = C\,\mathbf{1}(i{=}0) + \beta\, W(i) + \beta\lambda/r\,[W(i-1) - W(i)] + \beta\mu/r\,[W(i+1) - W(i)] \qquad (5.1)$$

Let $\rho \triangleq \lambda/\mu$. It is easy to check that $W(i)$ can be expressed as:

$$W(i) = A\, a^i + A\varsigma\,(a/\rho)^{-i} \qquad (5.2)$$

$$a \triangleq \frac{1}{2}(1 + \rho + \frac{1-\beta}{\beta}\,\frac{r}{\mu}) - \left[\left(\frac{1}{2}(1 + \rho + \frac{1-\beta}{\beta}\,\frac{r}{\mu})\right)^2 - \rho\right]^{\frac{1}{2}}, \qquad (5.3)$$

$$A \triangleq C\,\{1 - \beta - \beta\mu/r\,(a-1) + [1 - \beta - \beta\,\mu/r((a/\rho)^{-1} - 1)\,]\,\varsigma\},\ \text{and} \qquad (5.4)$$

$$\varsigma \triangleq a^L\,(a/\rho)^L\,\frac{1 - \beta - \beta\,\lambda/r\,(a^{-1} - 1)}{1 - \beta - \beta\,\lambda/r\,(a/\rho - 1)}.$$

5.2 The Heuristic Rule

For the sake of a simpler notation we let A_1, A_2 and a denote the parameters which characterize the cost to go of Q_1 in isolation, and B_1, B_2 and b, the parameters which correspond to Q_2 in isolation. They are computed according to equations (5.3), (5.4) , using the appropriate numbers for λ and μ. In order to work on the same probability space of the original problem, we let the transition rate r be as defined by (2.2).

We propose to replace the expected cost to go in the optimal decision equation (3.2) by the cost to go of each queue in isolation in order to get an empirical, easy to compute index rule. The underlying reasoning is as follows: If (i,j) is the state of the system, and the server is committed to serve Q_1 forever, the expected cost to go $J_1(i,j)$ is:

$$J_1(i,j) = Aa^i + A\varsigma(a/\rho_1)^{-i} + C_2\,\frac{1}{1-\beta}\,\{\frac{\beta\,\lambda_2/r}{1-\beta+\beta\lambda_2/r}\}^j$$

By the same token, the expected cost to go if the server were dedicated to Q_2 forever would be:

$$J_2(i,j) = Bb^i + B\varsigma(b/\rho_2)^{-j} + C_1\,\frac{1}{1-\beta}\,\{\frac{\beta\,\lambda_1/r}{1-\beta+\beta\lambda_1/r}\}^i$$

If the discount factor β is not too close to 1, we could expect $J_1(i,j) \approx V(i,j)$ provided that (i,j) is well into the region Ω_1 (far away from the switching curve and from the line $j = 0$). Similarly, we could expect $J_2(i,j)$ to approximate $V(i,j)$ in the region which lies in Ω_2 and is far away from both the switching curve and the boundary $i = 0$.

We now make the promised substitution, and we obtain the heuristic rule:

$$\mu_1\left[A_1 a^i(1-a) + A_2(a/\rho_1)^{-i}(1-\rho_1/a)\right] \underset{Q_2}{\overset{Q_1}{\gtrless}} \mu_2\left[B_1 b^j(1-b) + B_2(b/\rho_2)^{-j}(1-\rho_2/b)\right] \qquad (5.9)$$

where $\rho_i = \dfrac{\lambda_i}{\mu_i}$, $i = 1,2$. This rule simplifies even further if the infinite model is considered, since in this case $A_2 = B_2 = 0$. Taking logarithms, we obtain in this case the following linear switching curve.

$$i \log a - j \log b \underset{Q_2}{\overset{Q_1}{\gtrless}} \log \frac{\mu_2 B_1 (1-b)}{\mu_2 A_1 (1-a)} \tag{5.10}$$

It is easy to verify that (5.10) partitions the decision space according to a switching curve with the following characteristics: i) It is a straight line. ii) For symmetric systems ($\lambda_1 = \lambda_2$ and $\mu_1 = \mu_2$) it is a 45^0 slope line. iii) It implies an *index rule* policy. As such, it suggests an immediate generalization to the multiqueue scenario.

These characteristics agree with the experimental observations made with respect to the optimal switching curve.

5.3 A Comparison Between the Heuristic Rule and the Optimal Rule.

In this sub-section we compare the heuristic rule to the optimal rule.

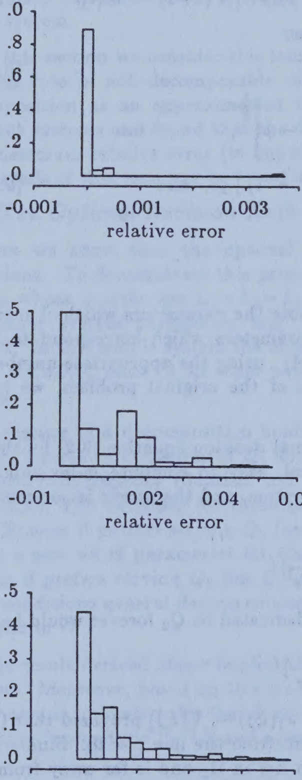


Figure 5.1: Varying Utilization.

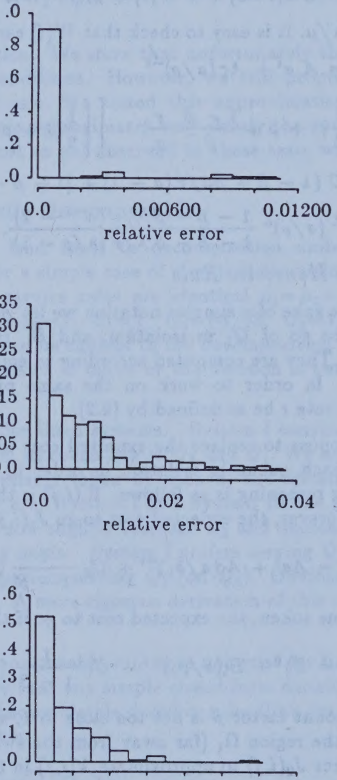


Figure 5.2: Fixed Total Utilization - 0.9.

In this comparison we examined a system consisting of three queues and varied the paramenters by letting them have the following values: 1) The parameters of Q_1 were set fixed to: $L_1 = 4$, $\mu_1 = 1$, $\lambda_1 = 1$ and $C_1 = 1$. 2) The service rate for all queues was set to be fixed - $\mu = 1$. 3) The cost parameters were varied by letting C_2 and C_3 independently have any of the values 0.1, 1, 5. 4) The arrival rate parameters were varied by letting λ_2 and λ_3 independently have any of the values 0.1, 1, 5. 5) The queue length were varied as to allow Q_2 and Q_3 having the following length combinations:

(2,2), (2,4), (2,9), (4,4), (4,9) and (9,9). This yields an experiment of 486 comparisons. In each of these tests we compared the heuristic rule to the optimal rule by examining the expected cost to go at the origin (namely, the point (0,0,0)). Note that this point is the one which is most likely to be sensitive to variations on the decision rule used. For each test we calculated the relative error between the expected cost to go of the optimal rule and the expected cost to go of the heuristic rule. A histogram of this comparison is given in Figure 5.1 where we plot the fraction of runs which achieved certain relative error. The figure has three parts: (a), (b) and (c) which correspond to three values of β: 0.9, 0.99 and 0.998, respectively.

Note that the accuracy achieved by the heuristic rule is pretty high: For $\beta=0.9$ the error of all runs was lower than 0.4%, and the majority of the runs (90%) had no error at all (namely, the heuristic rule was identical to the optimal rule!). Even for high values of β, e.g., $\beta=0.998$ the accuracy is still good where almost all runs (about 95%) have an error smaller than 5%.

While in the previous experiment we tried to be quite general, in the following one we investigate the quality of the heuristic rule in a more practical framework. Aware of the fact that most queueing systems are typically designed to operate in utilization of about 80-90%, we examine the heuristic rule under similar conditions. We do this by keeping $\mu=1$ and by having $\lambda_1+\lambda_2+\lambda_3=0.9$. The statistics for this comparison is summarized in Figure 5.2. The parameters used in this analysis are: 1) The parameters of Q_1 where set fixed to: $L_1=4$, $\mu_1=1$, and $C_1=1$. 2) The service rate for all queues was set to be fixed - $\mu=1$. 3) The cost parameters were varied by letting C_2 and C_3 independently have any of the values 0.1, 1, 5. 4) The arrival rate parameters were varied by letting λ_2 and λ_3 independently have any of the values 0.2, 0.3, 0.45, and by letting $\lambda_1=0.9-(\lambda_2+\lambda_3)$. 5) The queue length were varied as to allow Q_2 and Q_3 having the following length combinations: (2,2), (2,4), (2,9), (4,9) (for $\beta=0.9$ and $\beta=0.99$); For the experiment where $\beta=0.998$ the queue length combinations were (2,2), (2,4). The relative errors observed in this experiment are slightly higher than those observed in the previous one (Figure 5.1) but still in most cases it is lower than 5%.

REFERENCES

1. R. A. Milito and H. Levy, "Modeling and Dynamic Scheduling of a Queueing System with Blocking", working paper, AT&T Bell Laboratories, December 1986.
2. J.S. Baras, A.J. Dorsey and A.M. Makowski, "Two competing queues with linear costs and geometric service requirements: The μc -rule is often optimal", Adv. Appl. Prob., vol. 17, pp. 186-209, 1985.
3. C. Buyukkoc, P. Varaiya and J. Walrand, "The $c\mu$ Rule Revisited ", Adv. Appl. Prob., vol. 17, pp. 237-238, 1985.
4. J.C. Gittins, "Bandit processes and the dynamic allocation indices (with discussion)," J.R. Statist. Soc. B., vol. 41, pp. 148-164, 1979.
5. G.P. Klimov, "Time-Sharing Service Systems.I", Theory of Prob. and its Applications", vol. 19, # 3, pp. 532-551, 1974.
6. G.P. Klimov, "Time-Sharing Service Systems.II", Theory of Prob. and its Applications", vol. 23, # 2, pp. 314-321, 1978.
7. S. Stidham, "Optimal Control of Admission to a Queueing System", IEEE Trans. Autom. Control, Vol. AC-30, # 8, August 1985, pp. 705-713.
8. P. Varaiya, J. Walrand and C. Buyukkoc "Extensions of the Multiarmed Bandit Problem: The Discounted Case", IEEE Trans. Autom. Contr., Vol. AC-30, #5, May 1985, pp 426-439.
9. P. Whittle, "Optimization over Time", John Wiley & Sons, 1982.
10. S. Lippman, "Applying a New Device in the Optimization of Exponential Queueing Systems", Opns.Res. 23,687-710 (1975).
11. R. Serfozo, "An equivalence between discrete and continuous time Markov decision processes,", Oper. Res., vol.27,pp. 616-620, 1979
12. B. Hajek, "Optimal Control of Two Interacting Service Stations", IEEE Trans. Automat. Contr.,vol. AC-29, no.6, pp. 491-499, 1984.

STRUCTURAL DECOMPOSITION OF MULTIPLE TIME SCALE MARKOV PROCESSES*

J. R. ROHLICEK
BBN Laboratories Incorporated,
10 Moulton St., Cambridge, MA 02238.

A. S. WILLSKY
Laboratory for Information and Decision Systems,
Massachusetts Institute of Technology,
Cambridge, MA 02139.

Abstract

A straightforward algorithm for the multiple time scale decomposition of singularly perturbed Markov processes has been presented in [1,2]. That algorithm provides a uniform approximation of the probability transition function over the interval $t \geq 0$ through the construction of a sequence of aggregate models valid at successively slower time scales. When only the *structure* of these models is desired, the algorithm can be expressed simply in terms of graphs associated with each of the aggregated models. The major computation then becomes computing shortest paths in these graphs. This representation of the algorithm furthermore allows analysis of more complex systems where there are multiple perturbation parameters with unknown relative orders of magnitude.

Keywords: Markov processes, singular perturbation, multiple time scales, graph theory.

1 INTRODUCTION

In this paper, an algorithm for the analysis of the multiple time scale *structure* of a perturbed Markov process is developed which builds directly on the multiple time scale decoposition algorithm presented in [1,2]. Structure of a perturbed Markov process refers to the complete multiple time scale decomposition of the type performed in [1,2,3,4] where the evolution of the state probabilities is approximated using a sequence of aggregated models, each valid a successively slower time scales. In this case, however, only the position of the nonzero transition rates of each of the unperturbed, aggregated time scale models, and the sets of states which constitute the aggregate classes, are determined. Although the detailed behavior of the system cannot be recovered from these descriptions, much useful information is retained. It will be shown that this subset of information about the decomposition can be obtained using a very simple graph-theoretic algorithm which is implicit in the algorithm presented in [1,2].

Use of these structural aspects of a Markov chain can have many applications. For instance, an algorithm similar to that presented in this paper has been applied to the analysis of the behavior of "simulated annealing" optimization methods [5]. Other applications include computing order of magnitudes for the time of events in systems with rare transitions. Specific applications include analysis of the failure time of a fault-tolerant system or the error rate of a communication protocol.

The nature of the decomposition algorithms presented in this paper are very different from those which attempt to approximate the detailed behavior of system. In those algorithms, numerical calculations of dominant eigenvectors of unperturbed systems, and of ϵ-dependent "trapping" probabilities were required. The algorithms in this paper basically consist of computing various types of connectivity in labeled graphs. Since the computations involve only integer quantities, issues of numerical stability are not relevant. Also, by introducing the graph-theoretic formalism, it is possible to employ standard graphical algorithms for computing quantities such as shortest paths between vertices of a graph.

This paper is organized as follows. In the next section, the decomposition algorithm and uniform approximation result present in [1,2] is stated and a simple example is provided. In the next section, the graphical formalism is introduced and the new graphical decomposition algorithm is presented. The final section includes conclusions and a discussion of the relationship of this approach to other work. Also, the application of this algorithm to models with multiple perturbation parameters is discussed with application to a fault-tolerant system model.

---

*This research was conducted under the support of the Air Force Office of Scientific Research under grant AFOSR-82-0258 and the Army Research Office under grant DAAG-29-84-K005.

2 MARKOV PROCESS DECOMPOSITION ALGORITHM

In this section, the basic algorithm presented in [1,2] is stated. The reader is referred to those works for a detailed interpretation of this algorithm, proof of the uniform convergence of the approximation, and demonstration of its relationship to the algorithms of Courtois [4] and Coderch [3]. Following the algorithm, a simple example is provided.

Algorithm 1 (Continuous Time) *Begin with the generator*[1] $A^{(0)}(\epsilon)$ *of a finite-state Markov process with one ergodic class for* $\epsilon > 0$. $a_{ji}^{(0)}(\epsilon)$ *is the probability transition rate from state* i *to state* j. *Set* $k \leftarrow 0$.

1. *Partition the state set into the ergodic classes* $E_1, E_2, \ldots, E_N$ *and the transient set* T *generated by* $A^{(k)} = A^{(k)}(0)$. *If there is only a single class* $(N = 1)$, *the iteration is terminated.*
2. *For each class* E_I, *compute the ergodic probabilities* $u_{iI}^{(k)}$, $\forall i \in E_I$ *of the member states corresponding to the generator* $A^{(k)}$. *Also set* $u_{jI}^{(k)} = 0$, $\forall j \notin E_I$.
3. *For each state* j *and each class* E_I, *compute terms* $\tilde{v}_{Ij}^{(k)}(\epsilon)$ *such that*

$$\tilde{v}_{Ij}^{(k)}(\epsilon) \;=\; v_{Ij}^{(k)}(\epsilon)(1 + \mathrm{O}(\epsilon)) \tag{1}$$

$$\sum_{K=1}^{N} \tilde{v}_{Kj}^{(k)}(\epsilon) \;=\; 1 \tag{2}$$

where

$$v_{Ij}^{(k)}(\epsilon) \equiv \Pr\left(\eta^{(k)}(\epsilon, t^{*}) \in E_I \;\middle|\; \eta^{(k)}(\epsilon, 0) = j,\ t^{*} = \inf_{t \geq 0}(t \mid \eta^{(k)}(\epsilon, t) \notin T)\right) \tag{3}$$

and $\eta^{(k)}(\epsilon, t)$ *is a sample path of the Markov process generated by* $A^{(k)}(\epsilon)$.

4. *Form the* $n \times N$ *and* $N \times n$ *matrices*

$$U^{(k)} = \left[u_{iJ}^{(k)}\right] \quad \text{and} \quad \tilde{V}^{(k)}(\epsilon) = \left[\tilde{v}_{Ij}^{(k)}(\epsilon)\right] \tag{4}$$

Then

$$A^{(k+1)}(\epsilon) = \frac{1}{\epsilon}\tilde{V}^{(k)}(\epsilon)A^{(k)}(\epsilon)U^{(k)} \tag{5}$$

Set $k \leftarrow k + 1$. *Go to 1.*

The overall approximation of the evolution of the transition probabilities $\phi_{ji}^{(0)}(\epsilon, t)$ *(from state* i *to state* j*) can be written in matrix form as*

$$\begin{aligned}
\Phi^{(0)}(\epsilon, t) \;=\; & \Phi^{(0)}(t) + \\
& \left(U^{(0)}\Phi^{(1)}(\epsilon t)V^{(0)} - U^{(0)}V^{(0)}\right) + \\
& \left(U^{(0)}U^{(1)}\Phi^{(2)}(\epsilon^2 t)V^{(1)}V^{(0)} - U^{(0)}U^{(1)}V^{(1)}V^{(0)}\right) + \\
& \vdots \\
& \left(U^{(0)}\cdots U^{(k-1)}\Phi^{(k)}(\epsilon^k t)V^{(k-1)}\cdots V^{(0)} - \right. \\
& \qquad \left. U^{(0)}\cdots U^{(k-1)}V^{(k-1)}\cdots V^{(0)}\right) + \mathrm{O}(\epsilon)
\end{aligned} \tag{6}$$

where

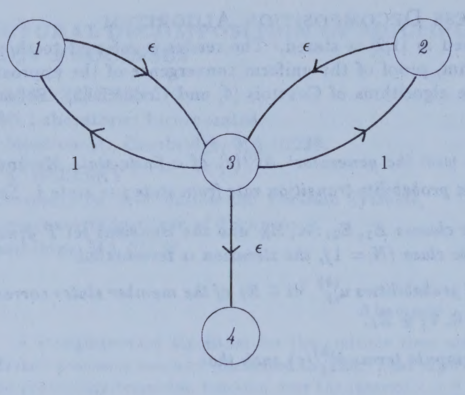
$$\Phi^{(k)}(\epsilon, \tau) = \mathrm{e}^{A^{(k)}(\epsilon)\tau},\ \Phi^{(k)}(\tau) = \mathrm{e}^{A^{(k)}\tau},\ \text{and } V^{(k)} = \tilde{V}^{(k)}(0) \tag{7}$$

and where $\mathrm{O}(\epsilon)$ *is a function of* ϵ *and* t *which converges uniformly to zero over* $t \geq 0$. □

A simple application of this algorithm is presented here. The same system will be analyzed using the graphical decomposition algorithm presented in the next section.

---

[1] The state probabilities $p(\epsilon, t)$ evolve according to $\frac{d}{dt}p(\epsilon, t) = A^{(0)}(\epsilon)p(\epsilon, t)$.



$$A^{(0)}(\epsilon) = \begin{bmatrix} -\epsilon & 0 & 1 & 0 \\ 0 & -\epsilon & 1 & 0 \\ \epsilon & \epsilon & -2-\epsilon & 0 \\ 0 & 0 & \epsilon & 0 \end{bmatrix}$$

Figure 1: Perturbed Markov process

Example 1 *Consider the generator of a Markov process*

$$A^{(0)}(\epsilon) = \begin{bmatrix} -\epsilon & 0 & 1 & 0 \\ 0 & -\epsilon & 1 & 0 \\ \epsilon & \epsilon & -2-\epsilon & 0 \\ 0 & 0 & \epsilon & 0 \end{bmatrix} \tag{8}$$

associated with the state transition diagram in Figure 1. The ergodic classes and transient set are

$$E_1 = \{1\},\ E_2 = \{2\},\ E_3 = \{4\},\ T = \{3\} \tag{9}$$

The ergodic probabilities are all degenerate in this case, therefore

$$u_{11}^{(0)} = u_{22}^{(0)} = u_{43}^{(0)} = 1 \quad or \quad U^{(0)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Suitable terms $\tilde{v}^{(0)}(\epsilon)$ which satisfy (1)-(2) above are

$$\begin{matrix} \tilde{v}_{13}^{(0)}(\epsilon) = \tilde{v}_{23}^{(0)}(\epsilon) = 1/2 - \epsilon/4 \\ \tilde{v}_{33}^{(0)}(\epsilon) = \epsilon/2 \end{matrix} \quad or \quad \tilde{V}^{(0)}(\epsilon) = \begin{bmatrix} 1 & 0 & 1/2-\epsilon/4 & 0 \\ 0 & 1 & 1/2-\epsilon/4 & 0 \\ 0 & 0 & \epsilon/2 & 1 \end{bmatrix} \tag{11}$$

Using these terms, $A^{(1)}(\epsilon)$ computed using (5) generates the process illustrated in Figure 2 and given by

$$A^{(1)}(\epsilon) = \begin{bmatrix} -1/2-\epsilon/2 & 1/2 & 0 \\ 1/2 & -1/2-\epsilon/2 & 0 \\ \epsilon/2 & \epsilon/2 & 0 \end{bmatrix} \tag{12}$$

The procedure is iterated to produce $A^{(1)}(\epsilon)$ and $A^{(2)}(\epsilon)$. $A^{(2)}(0)$ generates a single ergodic class and therefore the iteration is terminated. The set of ϵ-independent Markov models from which the approximation is derived is shown in Figure 3. □

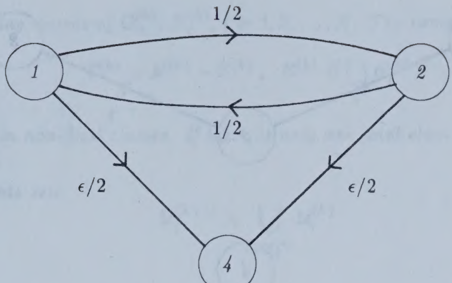


Figure 2: Order $1/\epsilon$ time scale model for Example 1

3 GRAPHICAL DECOMPOSITION

The uniform approximation (6) produced using Algorithm 1 is expressed as a combination of the behavior of a set of ϵ-independent, aggregated Markov processes with generators $A^{(0)}(0), \ldots, A^{(k)}(0)$ and some associated matrices (the $U^{(i)}$ and $V^{(i)}$) which characterize the recurrent and transient states of these processes. The algorithm can be greatly simplified if only the constituent states of the aggregate classes and the allowable (i.e. nonzero rate) state transitions are desired. In this section, the structure of the graphical representation is presented. Then, the graphical algorithm for the derivation of this entire structural decomposition is described.

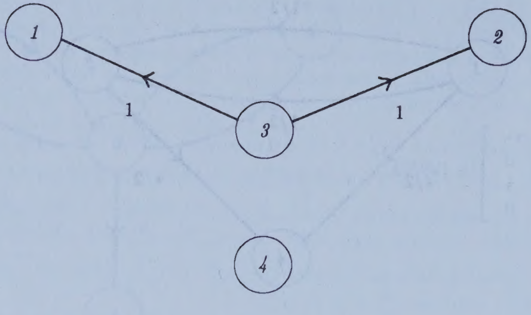
Given a perturbed Markov generator $A^{(k)}(\epsilon)$, an associated graph $G^{(k)} = (\mathcal{N}^{(k)}, \mathcal{E}^{(k)})$ can be constructed as follows. If $A^{(k)}(\epsilon)$ generates an n state process, then the set of vertices is $\mathcal{N}^{(k)} = \{1, 2, \ldots, n\}$. Each edge is a triple, $e = (i, j, w) \in \mathcal{E}^{(k)}$ of the initial vertex, i, the final vertex, j, and a nonnegative integer weight, w. For each nonzero entry $a_{ji}^{(k)}(\epsilon)$, $j \neq i$, a directed, weighted link from vertex i to vertex j is introduced. If $a_{ji}^{(k)}(\epsilon)$ is strictly $O(\epsilon^w)$, the weight of the edge is w. It is also useful to construct the "zero weight" graph $G_0^{(k)} = (\mathcal{N}^{(k)}, \mathcal{E}_0^{(k)})$ from the generator $A^{(k)}(0)$. The set of vertices is unchanged, and the new set of edges, $\mathcal{E}_0^{(k)} \subseteq \mathcal{E}^{(k)}$, corresponds to the subset of zero-weight edges in $\mathcal{E}^{(k)}$.

Several properties of the graph $G = (\mathcal{N}, \mathcal{E})$ will be used. First, define the distance between two vertices $d(i, j)$ as the minimum sum of weights along directed paths from i to j. If there is no such path, then $d(i, j) = \infty$. A *communicating class* C is a maximal set of states such that $i, j \in C$ if and only if $d(i, j) < \infty$. A class C is *final* if there is no vertex $j \notin C$ such that $d(i, j) < \infty$ for any $i \in C$. Also define the distance $\tilde{d}_E(i, C)$ as the minimum weight path from i to the class C with *no* intermediate vertices in the set E. There is a direct association between the Markov process generated by A and the graph G. Most important, an ergodic class generated by A exactly corresponds to a final class of G. Also, a transient state of A corresponds to a vertex of G in some non-final class.

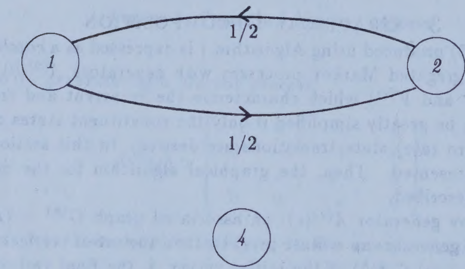
The basic structural decomposition algorithm can be specified in the graphical terms described above. We begin with a graph $G^{(0)}$ constructed using the Markov generator $A^{(0)}(\epsilon)$. Then a sequence of graphs associated with successive time scales, $G_0^{(0)}, G_0^{(1)}, \ldots, G_0^{(K)}$ is constructed such that $G_0^{(i)}$ is associated with the generator $A^{(i)}(0)$ constructed using Algorithm 1. Also, the constituent states in an aggregate class associated with each of the vertices in the graph are computed. The set $\mathcal{U}_i^{(k)}$ is the set of states of $\eta^{(0)}(\epsilon, t)$ associated with the aggregate class represented by vertex i in the graph $G_0^{(k)}$. The set $\mathcal{V}_i^{(k)}$ is the set of states which have an $O(1)$ probability of entering the aggregate class $\mathcal{U}_i^{(k)}$ by the k^{th} time scale.

Algorithm 2 (Structural Decomposition) *Begin with the graph $G^{(0)} = (\mathcal{N}^{(0)}, \mathcal{E}^{(0)})$ constructed from the Markov generator $A^{(0)}(\epsilon)$ of the n state process $\eta^{(0)}(\epsilon, t)$.*
Define $\mathcal{U}_i^{(0)} = \{i\}, \mathcal{V}_i^{(0)} = \{i\}$, $i = 1, 2, \ldots, n$.
Set $k \leftarrow 0$.

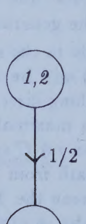
1. *Construct the graph $G_0^{(k)} = (\mathcal{N}^{(k)}, \mathcal{E}_0^{(k)})$ using the zero-weight subset of edges $\mathcal{E}_0^{(k)} \subseteq \mathcal{E}^{(k)}$. Identify*



(a) O(1) *time scale model — $A^{(0)}$*



(b) $O(1/\epsilon)$ *time scale model — $A^{(1)}$*



(c) $O(1/\epsilon^2)$ *time scale model — $A^{(2)}$*

Figure 3: Multiple time scale models for Example 1

the final communicating classes of $G_0^{(k)}$, $E_I^{(k)}$, $I = 1, 2, \ldots, N$. *The complementary set*

$$T^{(k)} \equiv \mathcal{N}^{(k)} - E^{(k)}\ , \quad E^{(k)} \equiv \bigcup_I E_I^{(k)} \tag{13}$$

is the set of vertices in non-final classes. If there is only one final class $(N = 1)$*, then the decomposition is completed.*

2. *Form the new aggregate sets*

$$\mathcal{U}_I^{(k+1)} = \bigcup_{i \in E_I^{(k)}} \mathcal{U}_i^{(k)} \tag{14}$$

3. *Define*

$$\bar{E}_I^{(k)} \equiv \{i \mid \tilde{d}_{E^{(k)}}(i, E_I^{(k)}) = 0\} \tag{15}$$

and form the new sets

$$\mathcal{V}_I^{(k+1)} = \bigcup_{i \in \bar{E}_I^{(k)}} \mathcal{V}_i^{(k)} \tag{16}$$

4. *Construct a new graph* $\bar{G}^{(k+1)} = (\mathcal{N}^{(k+1)}, \bar{\mathcal{E}}^{(k+1)})$, $\mathcal{N}^{(k+1)} = \{1, \ldots, N\}$. *For each pair of vertices* (I, J) *of* $\bar{G}^{(k+1)}$, $I \neq J$, *the following set of edges comprise* $\bar{\mathcal{E}}^{(k+1)}$.
   (a) *For each edge,* $e = (v_1, v_2, w) \in \mathcal{E}^{(k)}$, *where* $v_1 \in E_I^{(k)}$, $v_2 \in E_J^{(k)}$, *an edge* $(I, J, w - 1)$ *is added to the set* $\bar{\mathcal{E}}^{(k+1)}$.
   (b) *For each edge,* $e = (v_1, t, w) \in \mathcal{E}^{(k)}$, *where* $v_1 \in E_I^{(k)}$, $t \in T^{(k)}$, *such that* $\tilde{d}_{E^{(k)}}(t, E_J^{(k)}) < \infty$, *the edge* $(I, J, w - 1 + \tilde{d}_{E^{(k)}}(t, E_J^{(k)}))$ *is added to the set* $\bar{\mathcal{E}}^{(k+1)}$.
5. *The graph* $\bar{G}^{(k+1)}$ *is "reduced" by retaining only the lowest weight link between any pair of vertices producing the graph* $G^{(k+1)} = (\mathcal{N}^{(k+1)}, \mathcal{E}^{(k+1)})$. *Specifically,*
   (a) *Set* $\mathcal{E}^{(k+1)} \leftarrow \bar{\mathcal{E}}^{(k+1)}$.
   (b) *For each edge* $e = (v_1, v_2, w) \in \mathcal{E}^{(k+1)}$, *remove edge* e *if there is some other edge* $e' = (v_1, v_2, w'), w' \leq w$ *remaining in* $\mathcal{E}^{(k+1)}$

   Set $k \leftarrow k + 1$. *Go to step 1*

□

Note that the steps involved are numbered in a fashion to parallel the steps in Algorithm 1 with the exception that the last step is not performed in the previous algorithm. The major computational requirement of this algorithm involves determining the shortest paths, the $\tilde{d}$ terms. This can be accomplished using a variety of standard graphical algorithms. Since all shortest paths from non-final vertices to all final classes are required, an algorithm such as a modification of the Floyd algorithm [6] can be employed.

Algorithm 3 (Shortest path) *Begin with a graph* $G = (\mathcal{N}, \mathcal{E})$, $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{E} = \{(i, j, d_{ij}) \mid i, j \in \mathcal{N}\}$, *and a set of vertices* E *which may not be used as intermediated vertices.*

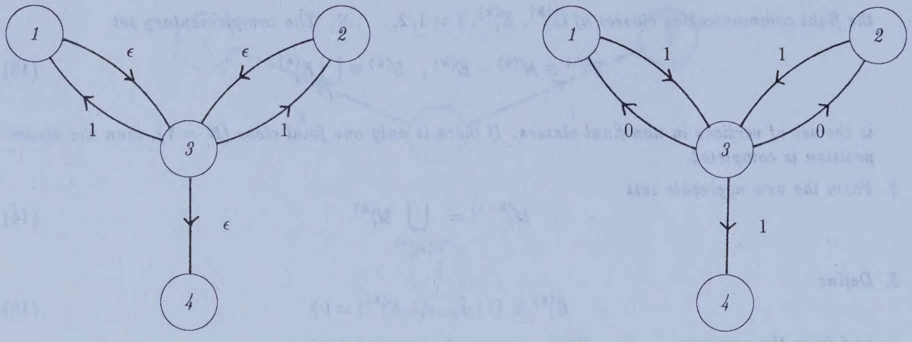
1. *Set* $D_{ij}^{(0)} = \min(d_{ij} \mid (i, j, d_{ij}) \in \mathcal{E})$. $D_{ij}^{(0)} = \infty$ *if there is no edge from* i *to* j.
2. *For* $n = 0, \ldots, N - 1$
   if $n \in E$ *then* $D^{(n+1)} = D^{(n)}$ *otherwise*

$$D_{ij}^{(n+1)} = \min\left(D_{ij}^{(n)}, D_{in}^{(n)} + D_{nj}^{(n)}\right) \quad \forall i \neq j$$

3. $\tilde{d}_E(i, j) = D_{ij}^{(N)}$

□

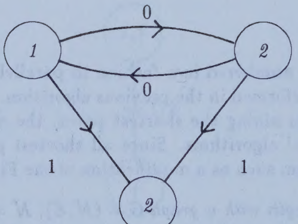
A simple example is considered using Algorithm 3 in order to clarify the approach. The Markov process considered appears in Example 1 where Algorithm 1 is demonstrated.


1
2
3
4
ε
ε
1
1
ε
1
2
3
4
1
1
0
0
1

(a) $A^{(0)}(\epsilon)$

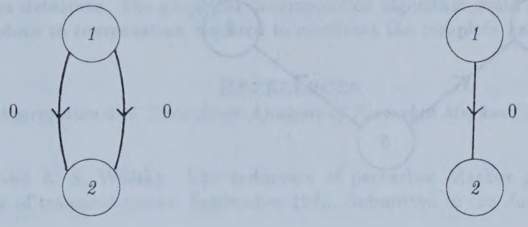
(b) $G^{(0)}$

Figure 4: Perturbed Markov process and associated graphs



$\bar{G}^{(1)} = G^{(1)}$

Figure 5: O(1/ϵ) time scale graph



(a) $\tilde{G}^{(2)}$ (b) $G^{(2)}$

Figure 6: $\mathrm{O}(1/\epsilon^2)$ time scale graphs

Example 2 *The generator $A^{(0)}(\epsilon)$ and the associated graph $G^{(0)}$ are shown in Figure 4. The graph $G_0^{(0)}$ has only the edges $(3,1,0)$ and $(3,2,0)$. The final classes of the graph $G_0^{(0)}$ are $E_1^{(0)} = \{1\}$, $E_2^{(0)} = \{2\}$, and $E_3^{(0)} = \{4\}$ and the non-final set is $T^{(0)} = \{3\}$. Using the definition in equation (15), $\tilde{E}_1^{(0)} = \{1,3\}$, $\tilde{E}_2^{(0)} = \{2,3\}$, and $\tilde{E}_3^{(0)} = \{4\}$. The distances in the graph $G^{(0)}$ to the final classes of $G_0^{(0)}$ are $\tilde{d}_{E^{(0)}}(3,1) = \tilde{d}_{E^{(0)}}(3,2) = 0$ and $\tilde{d}_{E^{(0)}}(3,4) = 1$. The sets $\mathcal{U}^{(1)}$ and $\mathcal{V}^{(1)}$ are $\mathcal{U}_1^{(1)} = \{1\}$, $\mathcal{U}_2^{(1)} = \{2\}$, and $\mathcal{U}_3^{(1)} = \{4\}$, and $\mathcal{V}_1^{(1)} = \{1,3\}$, $\mathcal{V}_2^{(1)} = \{2,3\}$, and $\mathcal{V}_3^{(1)} = \{4\}$.*

The graph $\tilde{G}^{(1)}$, which in this simple example is already reduced and therefore also the graph $G^{(1)}$, is shown in Figure 5. The sets constructed in this iteration on the algorithm ($k = 1$) are $E_1^{(1)} = \tilde{E}_1^{(1)} = \{1,2\}$, $E_2^{(1)} = \tilde{E}_2^{(1)} = \{3\}$, $T^{(1)} = \{\}$, $\mathcal{U}_1^{(2)} = \{1,2\}$, $\mathcal{V}_1^{(2)} = \{1,2,3\}$, and $\mathcal{U}_2^{(2)} = \mathcal{V}_2^{(2)} = \{4\}$.

Finally, the graph $\tilde{G}^{(2)}$ is constructed as shown in Figure 6(a). The reduction in this case corresponds to removing one of the duplicated links from vertex 1 to 2. The sets constructed in this iteration are $E_1^{(2)} = \{2\}$, $\tilde{E}_1^{(2)} = \{1,2\}$, and $T^{(2)} = \{1\}$. The set of graphs $G_0^{(i)}$ and the sets $\mathcal{U}^{(i)}$ and $\mathcal{V}^{(i)}$ are illustrated in Figure 7. This should be compared to the set of Markov processes illustrated in Figure 3 which is obtained using Algorithm 1. □

4 CONCLUSIONS

The graphical algorithm presented in this paper is significant in that it allows construction of an approximation of the behavior of a singularly perturbed Markov process in which the approximation preserves the multiple time scale structure of the system while not attempting to preserve the detailed dynamics at any time scale. Furthermore, there is no restriction on the ergodic structure of the process at any time scale. Specifically, there may be transient states at any of the intermediate time scale models.

An accurate approximation of the multiple time scale structure has many applications. In particular, complex systems which inherently have transient states can be analyzed using the new algorithm. Such models arise in the analysis of fault-tolerant systems. If failure of the system is associated with some combination of failures of critical components, then complicated multiple time scale structure may occur. The graphical algorithm presented in this paper provides a way to determine the critical sequences of events in the system which may lead to failure and the order of magnitude of the time till such events occur.

Another aspect of this algorithm which is introduced in [1] is that with a modest extension, the multiple time scale structure of systems with multiple perturbation parameters can be considered. Though perturbed Markov processes with two perturbation terms have been considered [7], that method requires making assumptions about the relative orders of magnitude of these parameters. In the extension of the graph algorithm, a set of multiple time scale structural decompositions can be derived where each decomposition is associated with a set of constraints on the relative orders of magnitudes of the perturbation terms. These constraints are associated with different paths in the graphs being associated with the shortest paths between sets of states.

Finally, the graphical algorithm allows identification of state transitions rates which are sufficiently small so that they cannot affect the full multiple time scale approximation formed using Algorithm 1. This observation would allow simplification of each of the generators $A^{(k)}(\epsilon)$ by setting certain transition

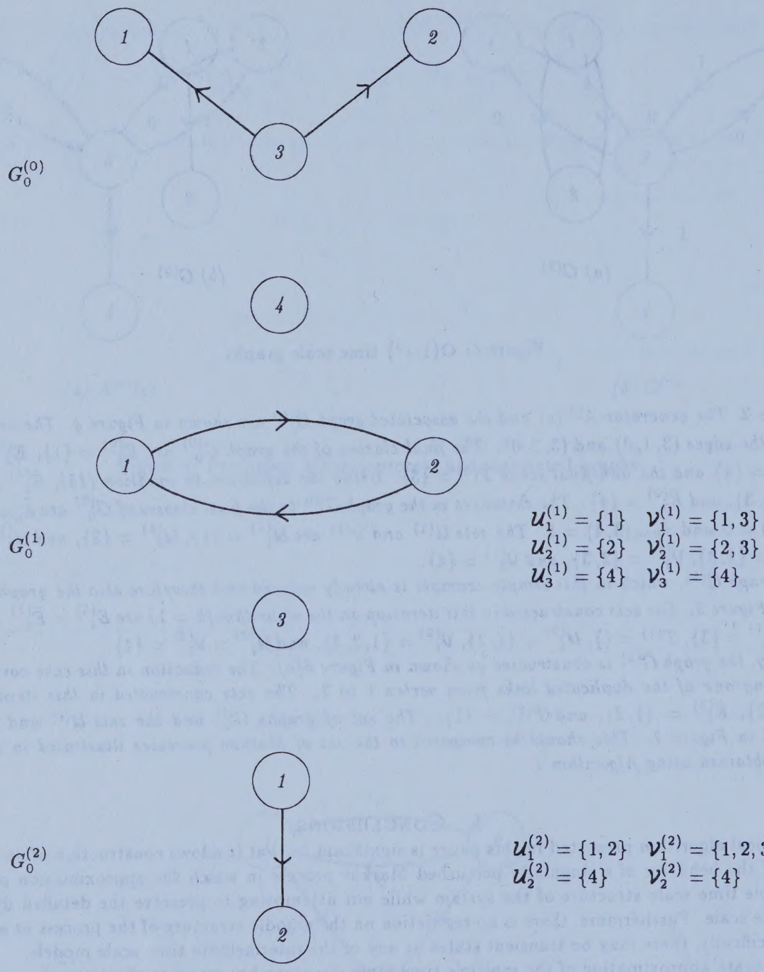

1
2
3
4
$G_0^{(0)}$
1
2
3
$G_0^{(1)}$
$\mathcal{U}_1^{(1)} = \{1\}$ $\mathcal{V}_1^{(1)} = \{1, 3\}$
$\mathcal{U}_2^{(1)} = \{2\}$ $\mathcal{V}_2^{(1)} = \{2, 3\}$
$\mathcal{U}_3^{(1)} = \{4\}$ $\mathcal{V}_3^{(1)} = \{4\}$
1
2
$G_0^{(2)}$
$\mathcal{U}_1^{(2)} = \{1, 2\}$ $\mathcal{V}_1^{(2)} = \{1, 2, 3\}$
$\mathcal{U}_2^{(2)} = \{4\}$ $\mathcal{V}_2^{(2)} = \{4\}$

Figure 7: Multiple time scale graphs

rates to zero between iterations. The graphical decomposition algorithm could in this way be used as an initial analysis to reduce to computation required to construct the complete uniform approximation.

REFERENCES

[1] J. R. Rohlicek. *Aggregation and Time Scale Analysis of Perturbed Markov Systems.* PhD thesis, MIT, 1987.

[2] J. R. Rohlicek and A. S. Willsky. The reduction of perturbed Markov generators: an algorithm exposing the role of transient states. September 1985. Submitted to the *Journal of the ACM.*

[3] M. Coderch, A. S. Willsky, S. S. Sastry, and D. A. Castinon. Hierarchical aggregation of singularly perturbed finite state Markov processes. *Stochastics*, 8:259–289, 1983.

[4] P. J. Courtois. *Decomposability: Queuing and Computer Systems Applications.* Academic Press, New York, 1977.

[5] J. N. Tsitsiklis. *Markov Chains with Rare Transitions and Simulated Annealing.* Technical Report LIDS-P-1497, MIT, September 1985.

[6] R. W. Floyd. Algorithm 97 — shortest path. *Communications of ACM*, 5:345, 1962.

[7] G. S. Ladde and D. D. Siljak. Multiparameter singular perturbations of linear systems with multiple time scales. *Automatica*, 19(4):385–394, 1983.

THE SETTLING TIME OF A CLOSED JACKSON NETWORK

V. ANANTHARAM

Cornell University, Phillips Hall, Ithaca, NY 14853, U.S.A.

ABSTRACT

In this paper we describe the transient behavior of a closed Jackson network. When there are N customers circulating, and the initial state is x, let $d_x^N(t)$ denote the total variation distance between the distribution at time t and the stationary distribution. Let $d^N(t) = \max_x d_x^N(t)$. We find a_N proportional to N, such that $d^N(t\, a_N) \rightarrow 1$ for *every* $t < 1$, and $d^N(t\, a_N) \rightarrow 0$ for *every* $t > 1$. Thus a_N can be interpreted as the settling time of the network, in that the network has not yet converged to stationarity upto this time but has converged to stationarity after this time.

I. INTRODUCTION

In this paper we describe the transient behavior of a closed Jackson network. In particular we identify, in a precise asymptotic sense, a notion of settling time for the network, which has the interpretation that the network has not yet converged to stationarity upto this time but has converged to stationarity after this time.

Consider a closed Jackson network, consisting of J exponential servers with service rates $\mu_1, \ldots, \mu_J$ and irreducible routing matrix $R = [r_{ij}]$, and with N customers circulating in the network. Recall that this means that a customer in queue at node i waits until all customers ahead of him have been served, and then receives service whose duration is an exponential random variable of mean μ_i^{-1}, all service times being independent. A customer finishing service at node i is routed to node j with probability r_{ij}, all routing decisions being independent and independent of the service times. Further, R is a nonnegative irreducible stochastic matrix. See Figure 1.

The network may be described by a Markov process $(X(t), t \geq 0)$ with state space $X = \{(n_1, \ldots, n_J) : \sum_{i=1}^{J} n_i = N\}$, where the state $x = (n_1, \ldots, n_J)$ corresponds to there being n_i customers in queue at node i. The stationary distribution of the network is well known, [7], [11], to be given by $\pi(n_1, \ldots, n_J) = c \prod_{i=1}^{J} (\frac{\lambda_i}{\mu_i})^{n_i}$, where c is a normalizing constant and λ_i, $1 \leq i \leq J$, are positive solutions of the flow balance equations $\lambda_i = \sum_{j=1}^{J} \lambda_j r_{ji}, \quad i = 1, \ldots, J$.

For a general irreducible continuous time Markov chain $(X(t), t \geq 0)$ with finite state space X and stationary distribution π, it is well known that whatever initial distribution the chain is started in, its

---

Research supported by the National Science Foundation, Grant No. NCR 8710840.

distribution will eventually converge to the stationary distribution in most senses of interest. Suppose the chain is started in state $x \in X$. A natural measure of the distance from stationarity at the time t is the total variation distance $d_x(t) = \frac{1}{2} \sum_{y \in X} | P_x(X(t) = y) - \pi(y) |$ between the distribution of the state at time t and the stationary distribution. The worst case distance from stationarity over all initial conditions, measured by $d(t) = \max_{x \in X} d_x(t)$ then provides a description of the transient behaviour of the chain. One knows that $d(t) \to 0$ as $t \to \infty$. In fact, if λ_2 denotes the second eigenvalue of the time 1 transition matrix of the Markov chain, one also knows that $d(t) \sim C\lambda_2^t$ as $t \to \infty$, for some constant C. Traditionally one thinks of λ_2 as describing the speed with which the chain approaches stationarity and of $-(\log\lambda_2)^{-1}$ as a time constant describing how long one has to wait before the chain has essentially converged.

The difficulty with considering this description satisfactory is that it is concerned only with the tail behavior of the convergence, and ignores the finite time behavior. This has been convincingly demonstrated by David Aldous, Persi Diaconis and their co-workers in a number of examples, [1], [2], [3], [6]. If λ_2 has large multiplicity, the qualitative behavior of $d(t)$ is somewhat like the graph in Figure 2. In particular, there appears to be a time τ such that $d(t)$ is very close to 1 upto τ, decreases rapidly near τ and then stays close to zero, with the tail being described in the traditional way, by the second eigenvalue λ_2. It is natural to think of τ as being the correct notion of settling time for the chain.

Returning to our closed network, consider the J numbers $\mu_i \lambda_i^{-1}$, $i = 1, ..., J$. When suitably scaled, $\mu_i \lambda_i^{-1}$ has the interpretation of the effective service rate at node i. Assume that there is a unique node with smallest effective service rate. Denote this node by s.

Let α_{ij} denote the probability that a customer leaving node i will meet node j at least once before returning to node i. Let w denote any node such that

$$\mu_w \alpha_{ws} - \mu_s \alpha_{sw} = \min_{i \neq s} \mu_i \alpha_{is} - \mu_s \alpha_{si}.$$

Let $a_N = \dfrac{N}{\mu_w \alpha_{ws} - \mu_s \alpha_{sw}}$. Let $d_x^N(t) = \frac{1}{2} \sum_{y \in X} | P_x(X(t) = y) - \pi(y) |$ and let $d^N(t) = \max_{x \in X} d_x^N(t)$ measure the worst case distance from the stationary distribution over all possible initial conditions, when there are N customers circulating in the network.

In this paper, we prove

Theorem 1.1: For any $t < 1$

$$\lim_{N \to \infty} d^N(t\, a_N) = 1 \tag{1.1}$$

whereas, for any $t > 1$

$$\lim_{N\to\infty} d^N(t\, a_N) = 0 \tag{1.2}$$

□

In the sense of Theorem 1.1, in the large N limit, the network has not yet converged to stationarity upto a_N but has converged to stationarity after a_N. This suggests that a_N should be considered the correct notion of settling time for the network.

While it is impossible to give all details in a report of this size, we have attempted to present enough information for the reader to reconstruct the proofs. Several of the proofs involve coupling, see [8]. For full details, we refer the reader to [4].

II. GENERALITIES

In this section we outline the general skeletal structure of the proofs of the succeeding sections. The methodology is that suggested by Aldous, [1].

To identify a threshold time, we first prove the lower bound (1.1) and then the upper bound (1.2). To prove the lower bound, fix $\delta > 0$, however small. We find a subset $K \subseteq X$ such that $\pi(K) \geq 1 - \delta$ and an element $x \in X$ such that the time to hit K starting from x is large. Namely, if $T = \inf\{t > 0 : X(t) \in K\}$, we prove that, for any $t < 1$

$$\lim_{N\to\infty} P_x(T \leq t\, a_N) = 0 \tag{2.1}$$

Note that x, K, X and π are really functions of N.

Note that $P_x(T \leq t\, a_N) \geq P_x(X(t\, a_N) \in K)$. Hence, if we can prove (2.1), it follows from the definitions that $d^N(t\, a_N) \geq d_x^N(t\, a_N) \geq |\, P_x(X(t\, a_N) \in K) - \pi(K)|$, for all sufficiently large N. Hence

$$\lim_{N\to\infty} d^N(t\, a_N) \geq 1 - \delta.$$

Letting $\delta \to 0$ establishes the lower bound.

To prove the upper bound, for any pair of initial states, $x, y \in X$, we couple the paths of the network started from x to those of the network started from y. This means that on the same sample space we construct two X valued Markov processes $(X^A(t), t \geq 0)$ and $(X^B(t), t \geq 0)$ such that $X^A(0) = x$ and $X^B(0) = y$ and such that (1) the A (respectively B) process is the same in law as the state process of the network started at x (respectively y) and (2) there is a proper random time $T_{x,y}$ after which the A and B processes agree, i.e.

$$X_i^A(t) = X_i^B(t) \text{ for all } 1 \leq i \leq J \text{ and } t \geq T_{x,y}.$$

Note that we do not require that $T_{x,y}$ is the first time at which the A and B processes agree, which is somewhat nonstandard. We call $T_{x,y}$ the coupling time.

Now let $\rho_{x,y}^N = \frac{1}{2} \sum_{z \in X} | P_x(X(t) = z) - P_y(X(t) = z)|$. Then it is easy to see that $d^N(t) \le \sup_{x,y} \rho_{x,y}^N(t)$. Further, for a coupling time $T_{x,y}$ it is easy to see that, for any $t \ge 0$, $\rho_{x,y}^N(t) \le P(T_{x,y} \le t)$. Now suppose that we can construct couplings for each pair of states $x, y \in X$ such that, for any $t > 1$

$$\lim_{N \to \infty} \sup_{x,y} P(T_{x,y} \le t\, a_N) = 0 .$$

Then by the above, we have, for any $t > 1$, $\lim_{N \to \infty} \sup_{x,y} \rho_{x,y}^N(t\, a_N) = 0$. Hence, for any $t > 1$,

$$\lim_{N \to \infty} d^N(t\, a_N) = 0$$

This establishes the upper bound.

The proofs of the main results in the following sections will conform to the above outline.

III. THE BASIC EXAMPLE

The basic example is the simple network of Figure 3, consisting of two nodes in tandem. Customers finishing service at a node join the queue at the other. There are N customers cicrulating in the system. Specializing Theorem 1.1, assume that the service rates are distinct and let $a_N = \frac{N}{\mu_w - \mu_s}$, where s denotes the slower node and w the faster. We prove Theorem 1.1 for this network. The basic intuition underlying the proof is that in a stable birth and death process the time taken to rise n steps is exponential in n, whereas the time taken to fall n steps is linear in n.

Instead of the standard two dimensional description, describe the network by a birth and death process with state space $X = \{0, 1, 2, \ldots, N\}$, having uprate μ_s and downrate μ_w. Let $\rho = \mu_s \mu_w^{-1} < 1$. The chain admits the stationary distribution $\pi(k) = c\rho^k$, $k = 0, 1, \ldots, N$, where c is a normalizing constant.

For the lower bound, note that for any $\delta > 0$, we can find $m > 0$ such that $\pi\{0 \le k \le m\} > 1 - \delta$ for all N. Start the network with $N - L$ customers in w, where $L = \log N$. Let

$$T = \inf\ \{t > 0 : X(t) \le m\}.$$

and

$$T_1 = \inf\ \{t > 0 : X(t) \le m \ \text{ or } \ X(t) = N\}.$$

Then $T_1 < T$.

ET_1 can be estimated by a simple martingale argument. To estimate $var\ T_1$, note that T_1 is stochastically dominated by the time to fall $N - m - L$ steps in a birth and death process without reflection, and use the resulting inequality :

$$ET_1^2 \leq N\sigma^2 + \frac{N^2}{(\mu_w - \mu_s)^2}.$$

Applying Chebyshev's inequality, we have

$$P(T \leq t\, a_N) \leq P(T_1 \leq t\, a_N) \leq \frac{\operatorname{var} T_1}{(ET_1 - t\, a_N)^2}. \tag{3.1}$$

If the estimates are done right, the right hand side of (3.1) can be seen to go to 0 as $N \to \infty$. The proof of the lower bound can now be completed by following the general reasoning in Section 2.

To establish the upper bound, for every pair of states $x, y \in X$, we couple the paths of the process starting from x to the paths starting from y. Since the state space is linear, the coupling is easy, and can be done by allowing the paths to evolve independently till they meet. One gets coupling times stochastically dominated by the time to fall N steps in a birth and death process without reflection, from which the upper bound follows by the general reasoning of Section 2.

□

IV. TANDEMS

Consider a closed tandem of J exponential servers with service rates $\mu_1, \ldots, \mu_J$ and with N customers circulating in the system, see Figure 4. Specializing the conditions of Theorem 1.1, assume that there is a unique slowest server, s, and let w denote one of the second slowest servers. Let $a_N = \dfrac{N}{\mu_w - \mu_s}$.

For the lower bound, note that for any $\delta > 0$, we can find $m > 0$ such that $\pi\{(n_1, \ldots, n_J) : n_s \geq N - m\} \geq 1 - \delta$, for all N. Start with all customers in w. The time till $N - m$ customers are in s exceeds the time till the number of customers in w falls to m. Call this T_1. To study T_1, pretend that the portion of network following node w is an initially empty open tandem of $J - 1$ nodes fed by the virtual departure process of w. Then T_1 is the first time the total number of customers in this tandem reaches $N - m$.

We now appeal to the *interchangeability* of a tandem of exponential servers, [5], [9], [10], [12]. This is the fact that if we consider an initially empty tandem of exponential servers being fed by an arbitrary arrival process, the law of the process of total number of customers in the system is the same whatever the order in which the servers are met. Rearrange the servers following w so that arriving customers first meet node s. T_1 is equal in distribution to $\tilde{T}_1$, the first time the total number of customers in the rearranged tandem reaches $N - m$.

The portion of the rearranged tandem following node s is stable even when fed by the virtual departure process of nodes s. Lemma A.1 applies. Hence, $\tilde{T}_1$ is essentially the time for the number of customers in s in the rearranged tandem to reach $N - m$. From the basic example, the lower bound follows.

To establish the upper bound, consider the portion of the network following s as an open tandem of $J-1$ nodes. Start this with N customers in the first node following s and fed with the virtual departure process of s. Let $\hat{T}$ denotes the first time the open tandem empties. Let $x=(n_1, \ldots, n_J)$ and $y=(m_1, \ldots, m_J)$ be any initial states for the original network, $\sum_{i=1}^{J} n_i = \sum_{i=1}^{J} m_i = N$. By identifying the virtual departure processes at the nodes, one can show with some work that $\hat{T}$ is a coupling time for the paths of the network started in x with the paths of the network started in y.

$\hat{T}$ is the time to empty for the open tandem following s started empty and with arrival process bringing N customers at time 0, and then the virtual departures of s. Apply interchangeability. $\hat{T}$ has the same distribution as T', the time to empty in the tandem where w is met first. The portion of the rearranged tandem following w is critical. Lemma A.3 applies. Hence T' is essentially the time for w to empty when it is started with N customers and is fed by the virtual departures of s. The upper bound follows by the basic example of Section 2.

□

V. GENERAL NETWORKS

We prove Theorem 1.1. For the lower bound, note that for any $\delta > 0$ we can find $m > 0$ such that $\pi\{(n_1, \ldots, n_J) : n_s \geq N-m\} \geq 1-\delta$, for all N. Start the network with $N-N^{4/5}$ customers in w and $N^{4/5}$ customers in s. The time till $N-m$ customers are in s exceeds T_1, the time till the number of customers in w falls to m. A rate calculation shows that the portion of the network other than the nodes w and s fed by the virtual departure processes of these nodes is a critical network. Lemma A.3 applies. Hence, on a time scale of order N, upto T_1 the portion of the network other than w and s is essentially punched through, upto fluctuations of size $N^{3/4}$. Further, s is essentially never empty on a time scale of order N. The lower bound follows from the basic example of Section 2.

For the upper bound, fix initial states x and y where there are N customers circulating. Consider the portion of the network other than s fed by the virtual departures of s. Call this the open process. For either initial condition, with the naive coupling of virtual departures, it can be seen that when this portion empties so does this portion in the original network, i.e., all N customers are in s. Run the open network processes from the initial conditions independently till both empty. Run the one which emptied first till it empties for the first time after the one which emptied last. We can stochastically upper bound the time between these empties by a time independent of N. Now couple excursions from the state with all N customers in s in the two processes in a one to one fashion, while letting the time spent in this state be independent exponentials in the two cases. The fluctuations in the exponentials will result in a coupling in a time stochastically upper bounded by a variable independent of N.

It remains to study the time T_x for the open process to empty from an initial state x with total number of customers at most N. We first show that it suffices to consider initial conditions where all N customers are in one of the queues. Having done this, for each node i we look at the subset of nodes which becomes unstable when the output of i is replaced by its virtual departures in the open process. By induction on the cardinality of this set and rate arguments we can finally show that for any $t > 1$

$$\limsup_{N \to \infty} \sup_x P(T_x < t\, a_N) = 1 .$$

From this the upper bound follows.

□

VI. COMPLEMENTS

In Theorem 1.1., we have assumed that there is a unique node of smallest effective service rate. It is easy to see that this assumption is necessary by considering the basic example of Section 3 when the two nodes have the same service rate. In this case the stationary distribution is uniform on $X=\{0, 1, \dots, N\}$, and $d^N(\frac{t}{N^2})$ can be seen to converge, as $N \to \infty$, to the total variation distance from stationarity for reflected Brownian motion on [0,1], started at 0. Note that this function is smooth, and the convergence is on a time scale of order N^2.

We can, however, generalize Theorem 1.1 to the situation when there is more than one node of smallest effective service rate, if we consider the distribution of the aggregated state which only counts the total number of customers in the subnetwork of nodes of smallest effective service rate.

Theorem 6.1 : Consider a closed Jackson network with J nodes, with notation as in Theorem 1.1. Suppose nodes $1, \dots, K$ are the nodes of smallest effective service rate, i.e., $\mu_k \lambda_k^{-1} = \min_{j=1,\dots,J} \mu_j \lambda_j^{-1}$ $k = 1, \dots, K$. For any node $i \in \{K+1, \dots, J\}$, let α^i_{pq}, $p,q \in \{1, \dots, K, i\}$ denote the probability that a customer leaving node p first returns to node q before visiting any of the other nodes in $\{1, \dots, K, i\}$. Let w denote any node such that $\mu_w(1 - \alpha^w_{ww}) - \sum_{k=1}^{K} \mu_k \alpha^w_{kw} = \min_i \mu_i(1 - \alpha^i_{ii}) - \sum_{k=1}^{K} \mu_k \alpha^i_{ki}$. Let

$$a_N = \frac{N}{\mu_w(1 - \alpha^w_{ww}) - \sum_{k=1}^{K} \mu_k \alpha^w_{kw}} . \tag{6.1}$$

Then the analog of Theorem 1.1 holds for the distribution of the aggregated state which ignores the detailed distribution of customers in the nodes $1, \dots, K$, with a_N as in (6.1).

□

APPENDIX

Three lemmas on the transient behaviour of open Jackson networks are central to the preceding analysis. Lemma A.1 and Lemma A.2 show that a stable open Jackson network is very much like a birth and death process. Lemma A.1 gives a sense in which the time to rise n steps is exponential in n. Lemma A.2 gives a sense in which the time to fall n steps is linear in n. Lemma A.3 complements Lemma A.1 for critical networks. It gives a sense in which the time to rise n steps in a critical network is proportional to n^2.

Lemma A.1 : Consider a stable open Jackson network consisting of J exponential servers with rates $\mu_1, \ldots, \mu_J$ respectively, outside world denoted by 0, exogenous Poisson arrivals of rate γ and $J+1 \times J+1$ routing matrix R. Let $(\lambda_i, i = 1, \ldots, J)$ solve the flow balance equations $\lambda_i = \gamma r_{oi} + \sum_{j=1}^{J} \lambda_j r_{ji}, \quad 1 \le i \le J$. Let s be any node such that $\mu_s \lambda_s^{-1}$ is the smallest of the J numbers $(\mu_i \lambda_i^{-1}, 1 \le i \le J)$. Suppose the network is started empty and let $(X_i(t), t \ge 0)$ denote the number of customers in queue at node i. Let $k', k > 0$ be such that $k' \log(\frac{\mu_s}{\lambda_s}) > k > 0$. Then

$$\lim_{N \to \infty} (\sum_{i=1}^{J} X_i(t) \le k' \log N \text{ for all } t \le N^k) = 1.$$

Proof : One first lower bounds the mean time to reach $k'\log N$, using an ergodic argument. This upper bounds the probability that one hits $k'\log N$ before returning to 0. Thinking of the time to hit $k'\log N$ as composed of a geometric number of returns to 0 followed by a large excursion gives the desired result.

□

Lemma A.2 : Consider a stable open Jackson network as in Lemma A.1. Let T_i, $1 \le i \le J$, have the distribution of the time to empty when the network is started with all queues empty except for a single customer in queue at node i. Let T^* have the smallest distribution which stochastically dominates all the T_i, $1 \le i \le J$. Suppose the network is started with n_i customers in queue at node i, where $\sum_{i=1}^{J} n_i = N$. Let T have the distribution of the time to empty. Then T is stochastically dominated by the sum of N independent copies of T^*.

Proof: The result follows by induction once one proves that the time to hit the set of states with total number of customers at most $N-1$ is stochastically dominated by T^*. This can be done by coupling.

□

Lemma A.3 : Consider an initially empty open Jackson network consisting of J exponential servers with service rates $\mu_1, \ldots, \mu_J$ respectively, $J+1 \times J+1$ routing matrix R, and exogenous Poisson arrivals of rate γ, see Lemma A.1. Suppose the network is critical, i.e. the solutions of the flow balance equations satisfy $\lambda_i \le \mu_i \quad i = 1, \ldots, J$. Then, for each $k', k > 0$ such that $2k' > k$, we have $\lim_{N\to\infty} P(\sum_{i=1}^{J} X_i(t) \le N^{k'} \quad \text{for all } t \le N^k) = 1$.

Proof: The basic intuition comes from the situation for a single critical M|M|1 queue. In the general case, one uses Lemma A.1 to show that the stable nodes can be considered punched through, leaving a set of critical nodes which behave essentially like M/M/1 queues.

□

REFERENCES

[1] D. Aldous, "Random Walks on Finite Groups and Rapidly Mixing Markov Chains," Preprint, 1987.

[2] D. Aldous and P. Diaconis, "Strong Uniform Times and Finite Random Walks", *Advances in Applied Mathematics, 8,* pp. 69-97, 1987.

[3] D. Aldous and P. Diaconis, "Shuffling Cards and Stopping Times", *American Mathematical Monthly, vol. 93,* no. 5, pp. 333-348, May 1986.

[4] V. Anantharam, "The Settling Time of a Closed Jackson Network", in preparation, 1987.

[5] V. Anantharam, "Probabilistic Proof of the Interchangeability of ./M/1 Queues in Series", to appear in *Queueing Systems,* 1988.

[6] P. Diaconis and M. Shahshahani, "Time to Reach Stationarity in the Bernoulli-Laplace Diffusion Model", *SIAM J. of Math Analysis, vol. 18,* no. 1, pp. 208-218, 1987.

[7] W. J. Gordon and G. F. Newell, "Closed Queueing Systems with Exponential Servers", *Operations Research, 15,* pp. 254-265, 1967.

[8] D. Griffeath, "A Maximal Coupling for Markov Chains", *Z. Wahr. ver. Geb., 31,* pp. 95-106, 1975.

[9] T. Lehtonen, "On the Ordering of Tandem Queues with Exponential Servers", *Jour. Appl. Prob., 23,* pp. 115 -129, 1986.

[10] P. Tsoucas and J. Walrand, "On the Interchangeability and Stochastic Ordering of ./M/1 Queues in Tandem", *Adv. Appl. Prob., 19,* No. 2, pp. 515 -520, 1987.

[11] J. Walrand, *An Introduction to Queueing Networks,* Prentice Hall Inc, 1987.

[12] R. Weber, "The Interchangeability of ./M/1 Queues in Series", *Jour. Appl. Prob., 16,* pp. 690 -695, 1979.

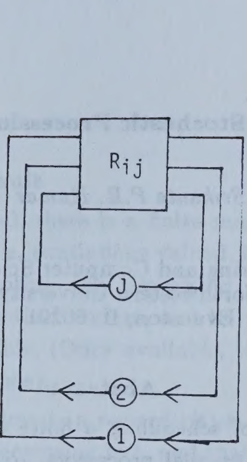


Figure 1

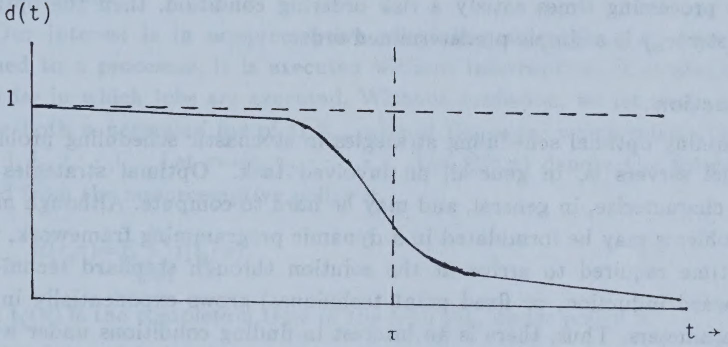


Figure 2

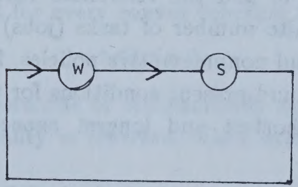


Figure 3

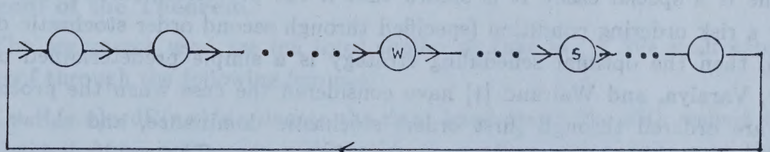


Figure 4

On Scheduling Tasks with Stochastic Processing Times on Parallel Servers.

Srikanta P.R. Kumar

Electrical Engineering and Computer Science Department
Northwestern University
Evanston, IL 60201

Abstract

We consider a problem of scheduling a finite number of jobs with random processing times on identical parallel processors, to maximize a reward criterion (minimizing flowtime is a special case). Scheduling is nonpreemptive. It is shown that if the processing times satisfy a risk ordering condition, then the optimal scheduling strategy is a simple predetermined order.

1. Introduction.

Determining optimal scheduling strategies in stochastic scheduling problems with parallel servers is, in general, an involved task. Optimal strategies are difficult to characterize, in general, and may be hard to compute. Although many of these problems may be formulated in a dynamic programming framework, typically the time required to arrive at the solution through standard techniques (e.g., backward induction, or fixed point techniques) grows exponentially in the problem parameters. Thus, there is an interest in finding conditions under which simple rules are obtained for the optimal scheduling strategy.

Several studies (see [1-5] and the references therein) have focused on the problem of scheduling a finite number of tasks (jobs) on parallel servers. These consider both preemptive and nonpreemptive policies, for optimizing criteria such as flowtime and makespan, and present conditions for which simple policies (such as SEPT and LEPT -- shortest and longest expected processing times) are optimal.

In this paper, we consider the problem of scheduling a finite number of tasks on identical parallel servers. Scheduling is nonpreemptive. Optimizing a discounted reward function is considered (in which the minimization of expected flowtime is a special case). It is shown that if the processing times of the jobs satisfy a risk ordering condition (specified through second order stochastic dominance), then the optimal scheduling strategy is a simple predetermined order. Weber, Varaiya, and Walrand [1] have considered the case when the processing times are ordered through (first order) stochastic dominance, and shown that SEPT is the optimal nonpreemptive scheduling policy. The result derived here extend the class of problems for which a simple optimal scheduling strategy is obtained.

2. Model and the Main Result

Suppose that, at time $t=0$, there is a finite number (n) of jobs that require service. Let X_i be a nonnegative, continuous valued, independent random variable (with finite expectation) denoting the processing (or service) time of job i, $1 \leq i \leq n$. There are $m(<n)$ identical servers, and let τ_i denote the time instant at which the i-th server is available. (Once available, the server may be utilized to execute any of the remaining jobs).

If a job is completed at time t, a reward $r(t)$ is received. It is assumed that both $r(t)$ and $\ddot{r}(t)$ are convex decreasing in t, and that $\dot{r}(t)$ is concave increasing in t. It may be noted that the usual form of discounting, $r(t)= e^{-\rho t}$, $\rho>0$, satisfies these conditions.

Our interest is in nonpreemptive scheduling strategies, i.e., once a job is assigned to a processor, it is executed without interruption. A strategy, thus, is the order in which jobs are executed. Without confusion, we let $\pi=(i_1,i_2, \cdots ,i_n)$ denote both a *permuted list* of $(1,2,...,n)$ and the *order* which selects the jobs in order $i_1,i_2, \cdots ,i_n$. Let $\tau=(\tau_1,\tau_2, \cdots ,\tau_n)$. Let $R(\tau,\pi)$ denote the total expected reward from the nonpreemptive policy π.

$$R(\tau,\pi)=E \sum_{k=1}^{n} r(t_k(\pi)) \tag{2.1}$$

where $t_k(\pi)$ is the completion time of the k-th job, under policy π.

Our interest is in characterizing the optimal policy π^* (ie., $R(\tau,\pi^*)\geq R(\tau,\pi)$) for the class of processing times which can be ordered through the following condition (2.2). We assume that for every convex function f,

$$E[f(X_i)]\geq E[f(X_{i+1})], \qquad 1\leq i\leq n-1 \tag{2.2}$$

In other words, job i has more risk or stochastically more variable than job $i+1$. (Note that the above inequality is reversed, when expectation with respect to a concave function is taken).

Theorem 2.1: In the above model, if processing times satisfy (2.2), then the optimal policy is the predetermined order $\pi^*=(1,2,...,n)$.

3. Proof of the Theorem.

The arguments used are (en toto) similar to that in [1]. We shall approach the proof through the following lemmas.

Let $R_i(\tau,\pi)\equiv dR(\tau,\pi)/d\tau_i$ denote the right hand derivative with respect to the arrival time of the i-th server. (With this convention, if in addition to the i-th server, other servers are available at τ_i, then jobs are first assigned to the other servers, according to π, and then finally to the i-th server. In other words, the i-th server may be thought of as being available slightly later). Similarly, let $R_{ii}(\tau,\pi)$ denote the (right hand) second derivative with respect to τ_i.

Lemma 3.1: Suppose there are n jobs which satisfy condition (2.2), and that the strategy $\pi^* = (1,2,\ldots,n)$ is followed. Then,

(a). For $j \neq i$, and $n \geq 1$, $R_{ii}(\tau,\pi^*)$ is decreasing in τ_i and increasing in τ_j, and

(b). Suppose $\pi_2^*=(2,3,\ldots,n)$, omitting some $k \geq 2$. Then for $n \geq 2$,

$$E[R_{11}(\tau_1+X_1,\tau_2+X_k,\ldots,\tau_m;\pi_2^*)-R_{11}(\tau_1+X_k,\tau_2+X_1,\cdots,\tau_m;\pi_2^*)] \geq 0.$$

Proof: The proof is by induction on n. Part (a) is trivial for n=1 (follows from the assumption that $\ddot{r}(t)$ is decreasing). Suppose part (a) is true where there are fewer than n projects. We show it holds for n projects also. Let i=1 without loss of generality. Also let $\tau_2 \leq \tau_3 \leq \cdots \leq \tau_m$. Consider three separate cases: (i) $\tau_1 < \tau_2$, (ii) $\tau_1 > \tau_2$, and (iii) $\tau_1 = \tau_2$. For the first case (defining $\pi_1^*=(2,3,\ldots,n)$),

$$R(\tau,\pi^*)=E[r(\tau_1+X_1)+R(\tau_1+X_1,\tau_2,\cdots,\tau_m;\pi_1^*)].$$

Hence,

$$R_{11}(\tau,\pi^*)=E[\ddot{r}(\tau_1+X_1)+R_{11}(\tau_1+X_1,\tau_2,\cdots\tau_m;\pi_1^*)].$$

It follows from the induction hypothesis and the assumption on $\ddot{r}(t)$, that the expression above over which the expectation is taken, is decreasing in τ_1; and, thus, proving part (a) for case (i).

For case (ii), notice that

$$R(\tau,\pi^*)=E[r(\tau_2+X_1)+R(\tau_1,\tau_2+X_1,\cdots\tau_m;\pi_1^*)].$$

and therefore,

$$R_{11}(\tau,\pi^*)=E[R_{11}(\tau_1,\tau_2+X_1,\cdots\tau_m;\pi_1^*)].$$

Again due to the induction hypothesis, part (a) is proved.

For case (iii), suppose that $\tau_2=\tau_3=\cdots=\tau_k<\tau_{k+1}\leq\cdots\leq\tau_m$. If $k>n$, then the jump in $R_{11}(\tau,\pi^*)$ when τ_1 takes the value τ_2 equals

$$E[\ddot{r}(\tau_2+X_k)-\ddot{r}(\tau_2+X_1)]\leq 0$$

If $k \leq n$, then the above jump equals

$$E[-\ddot{r}(\tau_2+X_k)]\leq 0.$$

This completes the proof to show that $R_{ii}(\tau,\pi^*)$ is decreasing (and also convex, which is established by looking at the third derivative in the above arguments) in τ_i. Similar arguments (omitted here) establish that this quantity is increasing in $\tau_j, j \neq i$.

To establish part (b), note that the claim holds for n=2 (for this case π_2^* is empty). The inductive step can be established by letting $h(x_1,x_2)=R_{11}(\tau_1+x_1,\tau_2+x_2,\ldots,\tau_m;\pi^*)$, and using the fact that this function is convex in x_1 and concave in x_2. This completes the proof of Lemma 3.1.

Lemma 3.2: Suppose the optimal order $\pi^*=(1,2,\ldots,n)$ is followed. Let $\pi_1^*=(2,3,\ldots,n)$. Then for $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_m$,

$$R_{11}(\tau,\pi^*) \geq E[R_{11}(\tau_1,\tau_2+X_1,\cdots,\tau_m;\pi_1^*)].$$

Proof: The proof is by induction on n. The lemma is obviously true for $n=1$ (note right side is zero, and left side is nonnegative). Suppose Lemmas is true for fewer than n projects. Let $\pi_2^*=(3,4,\ldots,n)$. Then

$$\begin{aligned}R_{11}(\tau,\pi) &= E[\ddot{r}(\tau_1+X_1)+R_{11}(\tau_1+X_1,\tau_2,\cdots,\tau_m;\pi_1^*)]\\ &\geq E[\ddot{r}(\tau_1+X_1)+R_{11}(\tau_1+X_1,\tau_2+X_2,\cdots,\tau_m;\pi_2^*)]\\ &\geq E[\ddot{r}(\tau_1+X_2)+R_{11}(\tau_1+X_2,\tau_2+X_1,\cdots,\tau_m;\pi_2^*)]\\ &= E[R_{11}(\tau_1,\tau_2+X_1,\cdots,\tau_m;\pi_1^*)].\end{aligned}$$

The first inequality follows from the induction hypothesis. The second inequality follows from part (b) of Lemma (3.1), and the assumptions on $\ddot{r}(t)$.

Proof of the Theorem: The proof is by induction on n. The theorem is obviously true for $n=1$. Suppose it is true for fewer than n jobs. The induction step is established through an interchange argument and contradiction.

Consider a scheduling order $\pi \neq \pi^*$ which assigns job $k(>1)$ at time τ_1. By the induction hypothesis, it is optimal to start job 1 next (at τ_2) and thereafter follow the rule in the theorem. Thus amongst strategies which start job k first, the best order is $\pi_{k,1} \equiv (k,1)+\pi_2^*$ which denotes the order $(k,1,\pi_2^*)$. Let $\pi_2^* = (2,3,\ldots,n)$ omitting job k. We shall show that the order $\pi_{1,k} \equiv (1,k)+\pi_2^*$ is better and thus completing the induction step. Let $R(\tau,\pi_{1,k};c)$ denote the expected reward from $\pi_{i,k}$, conditional on $X_k=c$. Define

$$\Delta(c)=R(\tau,\pi_{1,k};c)-R(\tau,\pi_{k,1};c).$$

To prove the theorem, it is enough to establish that $\Delta(c)$ is concave in c. For, if this is assumed, then

$$R(\tau,\pi_{1,k})-R(\tau,\pi_{k,1})=E[\Delta(X_k)] \geq E[\Delta(\overline{X}_1)]=0, \tag{3.1}$$

where $\overline{X}_1$ denotes an independent random variable having the same distribution as X_1. The equality to zero follows from the fact that if jobs 1 and k were identical then both strategies would yield the same expected reward. (3.1) would, then, complete the necessary step. In the following, we establish the concavity of $\Delta(c)$. It is readily seen that,

$$\begin{aligned}R(\tau,\pi_{1,k};c) = E[&I(X_1<\tau_2-\tau_1)\{r(\tau_1+X_1)+r(\tau_1+X_1+c)+R(\tau_1+X_1+c,\tau_2,\cdots,\tau_m;\pi_2^*)\}\\ &+I(X_1 \geq \tau_2-\tau_1)\{r(\tau_1+X_1)+r(\tau_2+c)+R(\tau_2+c,\tau_1+X_1,\cdots,\tau_m;\pi_2^*)\}].\end{aligned}$$

Also,

$$R(\tau,\pi_{k,1};c)=r(\tau_1+c)+R(\tau_1+c,\tau_2,\cdots,\tau_m;(1)+\pi_2^*).$$

Differentiating the above equations twice with respect to c, and making use of

Lemma 3.2, we can write

$$
\begin{aligned}
d^2\Delta(c)/d^2c \leq E[&I(X_1<\tau_2-\tau_1)\{\ddot{r}(\tau_1+X_1+c)+R_{11}(\tau_1+X_1+c,\tau_2,\cdots,\tau_m;\pi_2^*)\}+\\
&+I(X_1\geq\tau_2-\tau_1)\{\ddot{r}(\tau_2+c)+R_{11}(\tau_2+c,\tau_1+X_1,\cdots,\tau_m;\pi_2^*)\}-\\
&-\{\ddot{r}(\tau_1+c)+R_{11}(\tau_1+c,\tau_2+X_1,\cdots,\tau_m;\pi_2^*)\}].\\
\leq 0.&
\end{aligned}
$$

The last step follows from part (a) of Lemma 3.2, the fact that $\tau_1 \leq \tau_2$, and the assumption that $\ddot{r}(t)$ is decreasing. This demonstrates the concavity of $\Delta(c)$, thus completing the proof of the theorem.

4. Remarks and Conclusions.

The result here holds for the problem of minimizing expected flowtime, as this problem is a special case of the reward maximization considered here, which is seen by letting $r(t)=-t$. In this case, however, the optimal strategy may not be unique.

Condition (2.2) may be verified directly from the distributions of the processing times. If $F_i(x)$ denotes the distribution of X_i, then (2.2) holds if all random variables have the same mean, and if $\int_0^y [F_i(x)-F_{i+1}(x)]dx \geq 0$ for all y. (See Rotschild & Stiglitz (1970), for this notion of increasing risk). This condition also implies that the variances are ordered, with mean being the same.

As in previous models (such as this), the theorem may not hold, in general, if arbitrary job arrivals are allowed, or if the number of servers is decreasing over time, or if there are due dates on job completions.

References

[1] Weber, R.R., P. Varaiya and J. Walrand, "Scheduling Jobs with Stochastically Ordered Processing Times on Parallel Machines to Minimize Expected Flowtime," *J. Appl. Prob.*, 23 (1986).

[2] Pinedo, M., G. Weiss, "Scheduling Stochastic Tasks on Two Parallel Processors," *Naval Research Logistics Quarterly.*, Vol. 26, No. 3, pp. 527-536 (1979).

[3] Glazebrook, K.D. "Scheduling Tasks with Exponential Service Times on Parallel Processors," *J. Appl. Prob.*, 16, pp. 685-689 (1979).

[4] Weiss, G. and M. Pinedo, "Scheduling Tasks with Exponential Service Times on Nonidentical Processors to Minimize Various Cost Functions," *J. Appl. Prob.*, 17, pp. 187-202 (1980).

[5] Weber, R.R., "Scheduling Jobs with Stochastic Processing Requirements Parallel Machines to Minimize Makespan or Flow time," *J. Appl. Prob.*, 19,

pp. 167-182 (1982).

[6] Rothschild M. and Stiglitz, J., "Increasing Risk: A Definition," *J. of Economic Theory*, 9, pp. 185-202, (1970).

GLOBAL CONVERGENCE OF RECURSIVE STOCHASTIC MINIMIZATION ALGORITHMS

S. GELFAND
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

S.K. MITTER
Center for Intelligent Control Systems
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

We study the convergence of recursive stochastic algorithms of the form $X_{k+1} = X_k - a_k(\nabla U(X_k) + \xi_k) + b_k W_k$ by considering an *associated stochastic differential equation* $dx(t) = -\nabla U(x(t))dt + c(t)dw(t)$ whose convergence properties have recently been determined [5]-[8]. Here $U(\cdot)$ is a smooth function on a compact subset of $\mathbb{R}^r$, $\{\xi_k\}$ is a sequence of $\mathbb{R}^r$-valued random variables, $\{W_k\}$ is a sequence of independent standard r-dimensional Gaussian random variables, and $w(\cdot)$ is a standard r-dimensional Wiener process. We show under suitable conditions on $\{\xi_k\}$, $\{a_k\}$, and $\{b_k\}$ that X_k converges in probability to the set of global minima of $U(\cdot)$. Our results allow for $\{\xi_k\}$ with unbounded variance.

1. INTRODUCTION

It is desired to select a parameter value x^* which minimizes a smooth function $U(x)$ over $x \in D$, where D is compact subset of $\mathbb{R}^r$. If $U(\cdot)$ is unknown but can be measured with noise, then a Kiefer-Wolfowitz stochastic approximation algorithm [1] is often used:

$$Y_{k+1} = Y_k - a_k(\nabla U(Y_k) + \xi_k) , \tag{1.1}$$

where $\{\xi_k\}$ is a sequence of random variables which might arise, for example, from noisy finite difference approximations to $\{\nabla U(Y_k)\}$, and $\{a_k\}$ is a sequence of positive numbers with $a_k \to 0$ and $\sum a_k = \infty$. If $\{\xi_k\}$ is unbounded (and in many interesting cases it is and even has unbounded variance), it becomes necessary to bound the trajectories of $\{Y_k\}$; this may be done either by projecting Y_k back into D if it ever leaves D, or by fixing the dynamics in (1.1) so that Y_k never leaves D or only leaves D finitely many times w.p.1. Let S be the set of local minima of $U(\cdot)$ and S^* the set of global minima of $U(\cdot)$. Under suitable conditions on $U(\cdot)$, $\{\xi_k\}$ and $\{a_k\}$, and assuming that $\{Y_k\}$ is bounded, it is well-known that $Y_k \to S$ w.p.1. but in general $Y_k \not\to S^*$ w.p.1. or otherwise. Let $\mathcal{G}_k$ be the smallest σ-field which measures $\{Y_0, \xi_0, \ldots, \xi_{k-1}\}$. In particular, if $U(\cdot)$ is well-behaved, $a_k = A/k$ for k large where $A > 0$, and $E\{|\xi_k|^2 | \mathcal{G}_k\} = O(a_k^\alpha)$ and $E\{\xi_k | \mathcal{G}_k\} = O(a_k^\beta)$ as $k \to \infty$ where $\alpha > -1$ and $\beta > 0$, then $Y_k \to S$ w.p.1; see [2], [3] for precise conditions on $U(\cdot)$ and alternative assumptions on $\{\xi_k\}$ and $\{a_k\}$ (the above assumptions are very strong).

The analysis of the convergence w.p.1 of $\{Y_k\}$ is usually based on the stability of an *associated ordinary differential equation* (ODE).

$$\dot{y}(t) = -\nabla U(y(t))$$

This approach was pioneered by Ljung [4] and further developed by Kushner and Clark [2], Metivier and Priouret [3], and others. Kushner and Clark also analyzed the convergence in probability of $\{Y_k\}$ by this method. However, although their theory yields much useful information about the asymptotic behavior of $\{Y_k\}$ under very weak assumptions, it fails to obtain $Y_k \rightarrow S^*$ in probability unless S is a singleton; see [2, p. 125].

Consider the following generalization of the Kiefer-Wolfowitz algorithm:

$$X_{k+1} = X_k - a_k(\nabla U(X_k) + \xi_k) + b_k W_k \tag{1.2}$$

where $\{W_k\}$ is a sequence of independent Gaussian random variables with zero-mean and identity covariance matrix, and $\{b_k\}$ is a sequence of positive numbers with $b_k \rightarrow 0$. The $b_k W_k$ term is added in artificially so that $\{X_k\}$ can avoid getting trapped in a strictly local minimum of $U(\cdot)$. In general $X_k \not\rightarrow S^*$ w.p.1. (for the same reasons that $Y_k \not\rightarrow S^*$ w.p.1). However, under suitable conditions on $U(\cdot)$, $\{\xi_k\}$, $\{a_k\}$ and $\{b_k\}$, and assuming that $\{X_k\}$ is bounded, we shall show that $X_k \rightarrow S^*$ in probability. Let $\mathscr{F}_k$ be the smallest σ-field which measures $\{X_0, \xi_0, \ldots, \xi_{k-1}, W_0, \ldots, W_{k-1}\}$. In particular, if $U(\cdot)$ is well-behaved, $a_k = A/k$ and $b_k^2 = B/k \log\log k$ for k large where $B/A > C_0$ (a positive constant which depends only on $U(\cdot)$), and $E\{|\xi_k|^2 | \mathscr{F}_k\} = O(a_k^\alpha)$ and $E\{\xi_k | \mathscr{F}_k\} = O(a_k^\beta)$ as $k \rightarrow \infty$ where $\alpha > -1$ and $\beta > 0$, then $X_k \rightarrow S^*$ in probability.

Our analysis of the convergence in probability of $\{X_k\}$ is based on the stability of what we will call the *associated stochastic differential equation* (SDE)

$$dx(t) = -\nabla U(x(t))dt + c(t)dw(t) \tag{1.3}$$

where $w(\cdot)$ is a standard r-dimensional Wiener process and $c(\cdot)$ is a positive function with $c(t) \rightarrow 0$ as $t \rightarrow \infty$ (take $t_k = \sum_{n=0}^{k-1} a_n$ and $b_k = \sqrt{a_k}\, c(t_k)$ to see the relationship between (1.2) and (1.3)). There has been intense interest in using $x(\cdot)$ for global optimization (this has been called continuous-state simulated annealing); see [5]-[8]. The following result has been proven: if $U(\cdot)$ is well-behaved and $c^2(t) = C/\log t$ for t large where $C > C_0$ (a positive constant which depends only on $U(\cdot)$) then $x(t) \rightarrow S^*$ in probability as $t \rightarrow \infty$.

Kushner [8] also considers (1.2) for the case of $a_k = b_k = A/\log k$, k large. Although Kushner obtains a detailed asymptotic description of $\{X_k\}$ for this case, in general $X_k \not\rightarrow S^*$ in probability unless $\xi_k = 0$. The reason for this is intuitively clear: even if $\{\xi_k\}$ has bounded variance $a_k \xi_k$ and $a_k W_k$ will be of the same order and can interfere with each other. On the other hand by considering (1.2) for the case of $a_k = A/k$, $b_k^2 = B/k \log\log k$, k large, we get $X_k \rightarrow S^*$ in probability for $\{\xi_k\}$ with *unbounded* variance, in particular for $E\{|\xi_k|^2\} = O(k^\gamma)$ and $\gamma < 1$. Kushner's analysis of $\{X_k\}$ is similar to his analysis of $x(\cdot)$; he does not obtain the asymptotic behavior of $\{X_k\}$ from $x(\cdot)$ as we do.

2. MAIN RESULTS

Our analysis, like Kushner's [8], requires that we bound the trajectories of $\{X_k\}$. We proceed as follows. Take D to be a sphere centered at the origin. The actual algorithm is

$$\tilde{X}_{k+1} = X_k - a_k(\nabla U(X_k) + \xi_k) + b_k\sigma(X_k)W_k$$

$$X_{k+1} = \begin{cases} \tilde{X}_{k+1} & \text{if } \tilde{X}_{k+1} \in D \\ X_k & \text{if } \tilde{X}_{k+1} \notin D\ , \end{cases} \tag{2.1}$$

and the associated SDE is

$$dx(t) = -\nabla U(x(t))dt + c(t)\sigma(x(t))dw(t)\ .$$

We shall make assumptions which will force $\{\tilde{X}_k\}$ and $x(\cdot)$ to eventually stay in D w.p.1.

Let D_1 be another sphere centered at the origin, with $D_1 \subset D$ (strictly). Consider the following assumptions:

(A1) $U(\cdot)$ is a twice continuously differentiable function from D to $[0,\infty)$ with $\min_{x \in D} U(x) = 0$ and $(\nabla U(x),x) < 0$ for all $x \in D \backslash D_1$.

(A2) $\sigma(\cdot)$ is a Lipshitz continuous function from D to $[0,1]$ with $\sigma(x) > 0$ for all $x \in D\backslash \partial D$, $\sigma(x) = 1$ for all $x \in D_1$, and $\sigma(x) = O(d(x,\partial D))$ as $x \to \partial D$ $(d(x,y) = |x-y|)$.

(A3) $\{\xi_k\}$ are zero in $D\backslash D_1$ and $\{W_k\}$ are truncated in $D\backslash D_1$.

(A4) $a_k = \dfrac{A}{k}$, $b_k^2 = \dfrac{B}{k \log\log k}$, k large, where $A, B > 0$.

(A5) $c^2(t) = \dfrac{C}{\log t}$, t large, where $C > 0$.

(A6) $E\{|\xi_k|^2|\mathscr{F}_k\} = O(a_k^\alpha)$, $E\{\xi_k|\mathscr{F}_k\} = O(a_k^\beta)$ as $k \to \infty$ uniformly w.p.1, where $\alpha, \beta \in \mathbb{R}$.

For every $\epsilon > 0$ let

$$\pi^\epsilon(x) = \frac{1}{Z^\epsilon} \exp\left[-\frac{2\,U(x)}{\epsilon^2\sigma^2(x)}\right],\quad Z^\epsilon = \int_D \exp\left[-\frac{2\,U(x)}{\epsilon^2\sigma^2(x)}\right] dx$$

(A7) π^ϵ has a unique weak limit π as $\epsilon \to 0$.

Note that if we assume (A1), (A2), and (A7) hold, then it is clear that π concentrates on S^*, the global minima of $U(\cdot)$. The existence of π and a simple characterization in terms of the Hessian of $U(\cdot)$ is discussed in [9].

In practice (A1)-(A3) may be achieved as follows. Let D_2 be another sphere centered at the origin with $D_2 \subset D_1$ (strictly) and assume that $(\nabla U(x),x) < 0$ for all $x \in D\backslash D_2$; if not, add a penalty function to $U(\cdot)$. Now let $\phi(\cdot)$ be a smooth function from D to $[0,1]$ such that $\phi(x) = 1$ for $x \in D_2$ and $\phi(x) = 0$ for $x \in D\backslash D_1$, and define

$$\nabla\tilde{U}(x) = \phi(x)\nabla U(x) - (1-\phi(x))x \quad ; \quad \tilde{\xi}_k = \phi(X_k)\xi_k \,.$$

In practice we use

$$\nabla\tilde{U}(X_k) + \tilde{\xi}_k = \phi(X_k)\,(\nabla U(X_k) + \xi_k) - (1-\phi(X_k))X_k$$

in place of $\nabla U(X_k) + \xi_k$ in (2.1). Note that the global minima of $U(\cdot)$ and $\tilde{U}(\cdot)$ (which both lie in D_2) coincide.

If we assume that (A1), (A2), and (A5) hold then clearly $P\bigcap_{t\geq 0}\{x(t)\in D\} = 1$. We have the following result for $\{\tilde{X}_k\}$.

Proposition: Assume (A1)-(A4), (A6). Then

$$P\{\tilde{X}_{k+1} \notin D \,|\mathscr{F}_k\} = O(a_k^{2+\alpha}) \quad \text{as } k\to\infty\,,$$

uniformly w.p.1.

The proof of the Proposition is straight forward and may be found in [10]. We remark that by the Borel-Cantelli Lemma $P\bigcup_n \bigcap_{k\geq n} \{\tilde{X}_k \in D\} = 1$.

For a process $u(\cdot)$ and Borel function $f(\cdot)$, let $E_{t_1,u_1}(f(u(t)))$ denote conditional expectation with respect to $u(t_1) = u_1$ and let $E_{t_1,u_1;t_2,u_2}(f(u(t)))$ denote conditonal expectation with respect to $u(t_1) = u_1$ and $u(t_2) = u_2$. Also for a Borel measure $\mu(\cdot)$ and a Borel function $f(\cdot)$ let $\mu(f) = \int f d\mu$.

If we assume that (A1), (A2), (A5) and (A7) then by a slight modification of the main result of [7] we have that there exists a constant C_0 such that for $C > C_0$ and any bounded and continuous function $f(\cdot)$

$$\lim_{t\to\infty} E_{0,x}(f(x(t))) = \pi(f)$$

uniformly for $x\in D$. In [7] the constant C_0 is denoted by c_0 and has an interpretation in terms of the action functional for the perturbed dynamical system $dx^\epsilon(t) = -\nabla U(x^\epsilon(t))dt + \epsilon\sigma(x^\epsilon(t))dw(t)$. Here is our theorem on the convergence of $\{X_k\}$.

Theorem: Assume (A1)-(A4), (A6), (A7), and let $\alpha > -1$, $\beta > 0$, and $B/A > C_0$. Then for any bounded continuous function $f(\cdot)$

$$\lim_{k\to\infty} E_{0,x}(f(X_k)) = \pi(f)$$

uniformly for $x\in D$.

The proof of the theorem requires the following three Lemmas. Lemmas 1 and 2 constitute our main contributions; Lemma 3 is a slight modification of results in [7]. Let $\{t_k\}$ and $\tau(\cdot)$ be defined by

$$t_k = \sum_{n=0}^{k-1} a_n\,, \quad k = 0,1,\ldots,$$

$$\int_s^{\tau(s)} \frac{\log s}{\log u}\, du = s^{2/3}\,, \quad s \geq 0\,.$$

Lemma 1: Assume (A1)-(A6) and let $\alpha > -1$ and $\beta > 0$. Then there exists $\gamma > 1$ such that for any bounded continuous function $f(\cdot)$

$$\lim_{n\to\infty} \sup_{k\,:\,t_n \leqq t_k \leqq \gamma t_n} E_{0,x;n,y}\,(f(X_k)) - E_{t_n,y}(f(x(t_k))) = 0$$

uniformly for $x,y \in D$.

Lemma 2: Assume (A1), (A2), (A5). Then for any bounded continuous function $f(\cdot)$

$$\lim_{n\to\infty} \sup_{s\,:\,t_n \leqq s \leqq t_{n+1}} E_{t_n,y}(f(x(\tau(s)))) - E_{s,y}(f(x(\tau(s)))) = 0$$

uniformly for $y \in D$.

Lemma 3: Assume (A1), (A2), (A5), and let $C > C_0$. Then for any bounded continuous function $f(\cdot)$

$$\lim_{s\to\infty} E_{s,y}(f(x(\tau(s)))) - \pi^{c(s)}(f) = 0$$

uniformly for $y \in D$.

The proofs of these Lemmas may be found in [10]. We show here how they may be combined to prove the Theorem.

Proof of Theorem: Note that $\tau(s)$ is a strictly increasing function and $s + s^{2/3} \leqq \tau(s) \leqq s + 2s^{2/3}$. Hence for $k \gg 1$ one can choose s such that $t_k = \tau(s)$. Clearly $s < t_k$ and $s \to \infty$ as $k \to \infty$. Furthermore one can choose n such that $t_n \leqq s \leqq t_{n+1}$ and

$$t_n \leqq t_n + t_n^{2/3} \leqq t_k \leqq t_{n+1} + 2t_{n+1}^{2/3} \leqq \gamma t_n\ .$$

Clearly, $n < k$ and $n \to \infty$ as $k \to \infty$. Let $p(0,x;n,A) = P\{X_n \in A | X_0 = x\}$. Then we can write

$$E_{0,x}(f(X_k)) - \pi(f) = \int_D p(0,x;n,dy)\,(E_{0,x;n,y}(f(X_k)) - \pi(f)) \tag{2.2}$$

Now

$$\begin{aligned} E_{0,x;n,y}(f(X_k)) - \pi(f) = {} & E_{0,x;n,y}(f(X_k)) - E_{t_n,y}(f(x(t_k))) \\ & + E_{t_n,y}(f(x(\tau(s)))) - E_{s,y}(f(x(\tau(s)))) \\ & + E_{s,y}(f(x(\tau(s)))) - \pi^{c(s)}(f) \\ & + \pi^{c(s)}(f) - \pi(f) \to 0 \quad \text{as} \quad k \to \infty \end{aligned} \tag{2.3}$$

uniformly for $x,y \in D$ by Lemmas 1-3 and (A7). Combining (2.2) and (2.3) completes the proof $\square$.

Our Theorem, like Kushner's [8], requires that the trajectories of $\{X_k\}$ be bounded. However, the version of Lemma 3 in [7] applies with $D = \mathbb{R}^r$ (the convergence is uniform in a compact set), assuming certain growth conditions on $U(\cdot)$. Unfortunately, even if we assume these growth conditions hold and furthermore that the sequence of random variables $\{X_k\}$ is tight, we cannot obtain analogs of Lemmas 1 and 2 by our methods. On the other hand, bounding the trajectories of $\{X_k\}$ seems useful and even necessary in practice. The reason is that even with the specified growth conditions $|X_k|$ tends occasionally to very large values which leads to numerical problems in the simulation.

3. EXAMPLE

As an example of our results consider the case where ξ_k arises due to a noisy (two-sided) finite difference estimate of $\nabla U(X_k)$, with difference interval d_k, $d_k \to 0$. If $U(\cdot)$ is three times continuously differentiable, then (c.f. [2, p. 10]) there exists positive constants K, K_1, K_2 such that

$$K_1 \frac{1}{d_k^2} \leqq E(|\xi_k|^2 | \mathscr{F}_k) \leqq K_2 \frac{1}{d_k^2}, \qquad |E(\xi_k | \mathscr{F}_k)| \leqq K d_k^2 \qquad \text{w.p.1.}$$

If we take $a_k = A/k$, $b_k^2 = B/k \log \log k$ and $d_k = D/k^\delta$ for k large, then (A6) will be satisfied for $0 < \delta < 1/2$.

Another example, the details of which may be found in [10], illustrates how the continuous-state Metropolis algorithm with time-varying temperature (another version of simulated annealing) can be put in the form of (1.2). This yields the first convergence result we know of for this method.

In both of these examples $\{\xi_k\}$ has unbounded variance.

4. CONCLUSION

In conclusion, we show that for a large class of recursive stochastic minimization algorithms we can analyze convergence in probability to the global minima by examining the asymptotics of an associated SDE. This is analogous to analyzing the convergence w.p.1 to the local minima by examining the asymptotics of an associated ODE. Given the importance of the ODE method in analyzing recursive estimation, identification, and control algorithms, (c.f. [11], [12]), we believe our results will provide a new and useful tool in this area.

5. REFERENCES

[1] Wasan, M.T., *Stochastic Approximation,* Cambridge University Press, 1969.

[2] Kushner, H. and Clark, D., *Stochastic Approximation Methods for Constrained and Unconstrained Systems,* Applied Math. Science Series 26, Springer, Berlin, 1978.

[3] Metivier, M. and Priouret, P., "Applications of a Kushner and Clark lemma to general classes of stochastic algorithm," *IEEE Transactions on Information Theory,* Vol. IT-30, pp. 140-151, 1984.

[4] Ljung, L., "Analysis of recursive stochastic algorithms," *IEEE Transactions on Automatic Control,* Vol. AC-22, pp. 551-575, 1977.

[5] Geman, S. and Hwang, C.R., "Diffusions for global optimization," *SIAM J. Cntrl. Opt.,* 24, pp. 1031-1043, 1986.

[6] Gidas, B., "Global optimization via the Langevin equation," *Proc. IEEE Conference on Decision and Control,* 1985.

[7] Chiang, T.S., Hwang, C.R. and Shew, S.J., "Diffusion for global optimization in $\mathbb{R}^n$," *SIAM J. Cntrl. Opt.,* 1987.

[8] Kushner, H.J., "Asymptotic global behavior for stochastic approximation and diffusions with slowly decreasing noise effects: Global minimization via Monte Carlo," *SIAM J. Appl. Math.,* 47, pp. 169-185, 1987.

[9] Hwang, C.-R., "Laplaces method revisited: weak convergence of probability measures," Ann. Prob., 8, pp. 1177-1182, 1980.

[10] Gelfand, S. and S.K. Mitter, "Global convergence of recursive stochastic minimization algorithms," submitted for publication.

[11] Ljung, L. and Soderstrom, T., *Theory and Practice of Recursive Identification,* MIT Press, Cambridge, MA, 1983.

[12] Kumar, P.R. and Varaiya, P., *Stochastic Systems: Estimation, Identification and Adaptive Control,* Prentice Hall, NJ, 1986.

A GLOBAL OPTIMAL CONTROL SETTING FOR LARGE DEVIATIONS

EFTHIMIOS KAPPOS

Department of Electrical and Computer Engineering,
Northeastern University, Boston, MA 02115.

ABSTRACT

In this work, we provide a dynamical setting for large deviations that completes the global description of transition between attractors given in Wentzell and Freidlin [1]. We also use an optimal control approach to the associated variational problem.

In addition to making precise the nature of the 'compacta' in [1]–we assume them here to be asymptotically stable compact manifolds–we elucidate the global structure of the saddle critical elements. This permits us to make more exact statements about the nature of the exit paths: they leave every region of attraction through a neighborhood of the nearest saddle in the control sense.

1. INTRODUCTION

Large deviation theory ([1],[2]) has generated considerable interest recently for a number of reasons: it provides instance where it is possible to calculate (at least asymptotically) probabilistic quantities relating to stochastic process problems which, otherwise, would be intractable (this is a feature shared with the analogous theory in quantum mechanics, namely the WKB method, or semi-classical approximation, see [3],[4]). The theory also gives new prominence to variational methods: the evaluation of probabilistic quantities reduces to solving an associated optimization problem.

For processes described by **diffusions** (continuous-time Markov stochastic processes satisfying stochastic differential equations) many aspects of the theory are especially clear. In this paper we give a novel setting for large deviations that involves a cleaner global dynamical setting and an optimal control approach to the variational problem. In particular:

(a) the **stability** required of the drift vector field for large deviation paths to make sense is elucidated by limiting ourselves to a class of dissipative dynamics appropriate for a large deviation theory. This class is closely related to the class of Morse-Smale vector fields in a compact manifold. In particular, we find that the condition of dissipativeness implies the existence of global Lyapunov functions that are strictly decreasing away from the ω- and α-limit sets.

(b) the **variational problem** is posed as an optimal control problem with system dynamics naturally related to the stochastic differential system and a cost-of-control cost functional. This setting, we believe, is advantageous for large deviations and it yields a better qualitative understanding of the large deviation paths.

This paper draws on previous work ([2],[5],[6]). An effort will be made to present the main ideas at the danger of being occasionally mathematically imprecise.

2. DYNAMICAL SETTING

First, we want to specify a class of vector fields in $\mathbf{R}^n$ that have a global stability structure, in the sense that all of their trajectories are 'attracted' to a compact set (possibly composed of many components) that has locally dimension less than n. This *stability structure* seems necessary, since otherwise the concept of a large deviation path is hard to define (for example, when the dynamics are chaotic).

Intuitively, we expect the stochastic process, for small noise intensity, to spend most of its time near the attractors, occasionally making rare excursions away from them that may be enough to kick us out of one region of attraction and into another. Some attractors will be easier to get away from than others and the measure of this ease turns out to be the minimum cost of control for steering the path to the boundary of a region of attraction. This generalizes the case of a gradient vector field, where this cost is proportional to the potential difference between the local minimum and the lowest saddle around it (see [6]).

When an attractor is asymptotically stable, its region of attraction is an open set. By keeping the global structure of the dynamics simple, we are able to obtain results that are completely analogous to the case of gradient vector fields. For example, the optimal exit path leads through a saddle critical element. A difference we will find is that more than one Lyapunov function is needed to solve the problem globally (one for each attractor).

Globally, large deviation theory concerns itself with discovering the structure of these transitions between attractors. As is shown in [1], a Markov chain model is applicable and in the limit as ϵ goes to zero, the invariant measure concetrates on those attractors that are the most stable in the control sense.

By being precise about the global structure of the dynamics, we improve on [1], where only the attractors are specified and the global saddle structure—and hence the concept of region of attraction—is unclear.

We now specify the class of dissipative vector fields $\mathbf{D}(\mathbf{R}^n)$ as follows:

A vector field $b \in \mathbf{D}(\mathbf{R}^n)$ iff:

(i) b is a smooth vector field that gives a global flow (i.e. is complete).

(ii) b has a finite number of connected attractors that are asymptotically stable and are compact submanifolds of $\mathbf{R}^n$ of dimension strictly less than n.

(iii) The α– and ω–limit set behavior is simple: there is a finite number of saddle critical elements (equilibria and limit cycles), there are no other ω–limit sets and there hold global transversality conditions between their stable and unstable manifolds.

(iv) the no-cycle condition is satisfied: roughly, there are no saddle connections (see [2] and [8] for details).

It is a property of these vector fields that there exist global Lyapunov functions that are strictly decreasing along the trajectories of b, except on the attractors and the saddle limit cycles. Thus the stability structure is seen here to imply the existence of Lyapunov functions, which in turn are a condition of *dissipativeness* similar to that defined in Willems [7].

Note that the existence result can be proved using a purely geometrical argument. It exploits the foliation of the (open) region of attraction by the trajectories of the system dynamics to demonstrate the existence of Lyapunov surfaces.

Unlike the case of gradient vector fields, to every vector field in the class $\mathbf{D}(\mathbf{R}^n)$ there may correspond many different types of Lyapunov functions. Their classification is given in [10] and is based on the way their level sets sweep past the saddle critical elements. As we see in the next section, the quasi-potential functions of large deviation theory turn out to be Lyapunov functions as defined here.

3. LARGE DEVIATION BEHAVIOR IN AN OPTIMAL CONTROL SETTING

We consider diffusion processes x_t that satisfy:

$$dx_t = b(x_t)dt + \epsilon\sigma(x_t)dw_t \tag{1}$$

with the assumptions:

(i) the drift vector field $b \in \mathbf{D}(\mathbf{R}^n)$,

(ii) the diffusion σ is a smooth function from $\mathbf{R}^n$ to $\mathbf{R}^{n\times n}$ and is non-singular for all x with eigenvalues bounded away from zero.

(iii) the explosion time is infinite, almost surely (a faster than linear growth condition on b is sufficient—see [9]).

For ϵ small, we stay close, over any bounded time interval, to a trajectory of:

$$\dot{x_t} = b(x_t), \tag{2}$$

with probability 1, asymptotically as $\epsilon \downarrow 0$.

Given the structure of the dynamics of b, this means that, almost always, the trajectory of the diffusion will be in the neighborhood of an attractor K of b. A **large deviation path** is one that exits R_K, the region of attraction of K and results in a 'transition' between two attractors.

To describe the asymptotic behavior of these large deviation paths, we consider the *associated control system* :

$$\dot{x}_t = b(x_t) + \sigma(x_t)u \tag{3}$$

with the cost-of-control functional (suppose x_0 fixed and $x_T = x$):

$$J_x(u) = \frac{1}{2}\int_0^T |u_t|^2\, dt. \tag{4}$$

We can now state the main result of large deviation theory in the setting of this paper:

Theorem:

Fix an attractor K of b and let $\{\sigma_1, \ldots, \sigma_m\}$ be the set of saddle critical elements belonging to $\bar{R}_K$.

Then the function: $V(x) \equiv \inf_u J_x(u)$ defined in R_K is smooth and is a Lyapunov function there. If σ^* minimizes V over all saddles above, then, for any $\delta > 0$ and any $x \in R_K$:

$$\lim_{\epsilon \downarrow 0} P_x\{|x_{\tau_K^\epsilon} - \sigma^*| > \delta\} = 0, \tag{5}$$

where: $\tau_K^\epsilon \equiv \inf_t\{x_t \notin R_K\}$ is the first exit time from the region of attraction of K.

Also we have for the asymptotic exit time:

$$\lim_{\epsilon \downarrow 0} \epsilon^2 \ln E_x \tau_K^\epsilon = V(\sigma^*). \tag{6}$$

These results are in the process of being generalized to the case of singular diffusion matrix, under the condition of Lyapunov controllability given in [5] (see also [2]).

Note that in previous treatments, the exit problem involved a domain attracted to the stable attracting set, but otherwise arbitrary. A natural candidate for such a domain in our context is the inside* of a Lyapunov surface that is complete in the sense that it intersects all trajectories with K as their ω–limit set. In fact it is part of the proof of the above theorem to construct such surfaces that contain all but a thin neighborhood of the boundary of the region of attraction of K and then to show that the Lyapunov function that solves the optimal control problem achieves its minimum on it near the saddle σ^*.

---

* defined as follows: if S is a Lyapunov surface, then its inside is the set of all points in its attractor and all points of the form $\phi(x,t)$ for $x \in S$ and $t \geq 0$

The other parts of the theorem are not difficult: it is easy to see that our cost-of-control functional (4) is equivalent to the calculus-of-variations one in [1] and that the quasi-potential is a Lyapunov function.

4. CONCLUSIONS

We found it profitable to employ a dynamical systems approach for the problem of large deviation exits. Motivated by the gradient vector field case, we defined a class of dissipative dynamics that makes possible the generalization of many of the results of that case in a natural way.

Lyapunov functions are the natural generalizations of Morse functions in dynamical system theory and in our study too. A nice result is the fact that large deviation exits are from saddle critical elements.

REFERENCES

[1] A. Wentzell, M. Freidlin: *Random Perturbations of Dynamical Systems*, Springer, 1984.

[2] E. Kappos: *Large Deviations in Dissipative Dynamics: An Optimal Control Approach*, Electronics Research Laboratory Memo.No UCB/ERL M86/86, University of California, Berkeley, December 1986.

[3] V.P. Maslov: *Semi-Classical Approximations in Quantum Mechanics*, North-Holland, 1980.

[4] E. Kappos: *The Stationary Phase Method in Quantum Mechanics, Stochastic Mechanics and Relations to Large Deviation Theory*, work in progress.

[5] E. Kappos, S. Sastry: *Lyapunov Controllability and Global Optimal Control*, 25th Control and Decision Conference, Athens, Greece, 1986.

[6] E. Kappos: *Saddle Exits in Optimal Control Problems*, 26th Control and Decision Conference, Los Angeles, 1987.

[7] J. Willems: *Dissipative Dynamical Systems*, Arch. Rat. Mech. and Appl., **45**, 1972.

[8] C. Pugh, M. Shub: *The Ω–Stability Theorem for Flows*, Invent. Math., **11**, 1970, pp.150-158.

[9] R.Z. Has'minskii: *Stochastic Stability of Differential Equations*, Sijthoff and Noordhoff, 1980.

[10] E.Kappos: *Global Lyapunov Function Theory*, submitted to the IEEE Transactions on Circuits and Systems.

OPTIMAL CONTROL OF ITEMIZED PRODUCTION TO MEET UNCERTAIN DEMAND

J.B. RIBEIRO DO VAL
Universidade Estadual de Campinas - UNICAMP
FEE - Depto. de Telemática
13.081 - Campinas, SP, Brazil

ABSTRACT

In this paper a model for control and optimization in a problem in manufacturing systems is studied. In the situation we focus here one wishes to supply a stochastic demand for a number of identical items on time of requests, while the inventory level of finished items is made as low as reasonable in order to keep operational costs to a minimum. The model accounts for the fact that labour should be aggregate to raw materials and parts to make a finished item. For this purpose an item is considered to be finished when the accumulated commitment of the operational resources directed to the item reaches a certain level. We show that this problem can be posed in the framework of controlled Piecewise Deterministic Processes. A simple solution is obtained by solving recursively a differential equation and combining the solution with an algebraic equation in order to construct the value function for the problem.

INTRODUCTION

We address the control problem of a manufacturing system producing a single identical item in which a stochastic demand process should be matched by the production process. Each item is completed by the unit after a total commitment period of length τ. A finished piece is then added to the inventory which will supply random requests in time and in size for the commodity. The production rate (or "dedication rate") of the unit can be adjusted in accordance to the knowledge of demand so that, eventually items will be produced as a policy in time periods larger than τ. In the control problem one whishes to avoid surplus due to over production and also shortage due to miscalculated idle periods. The external variable is the production rate. It can be varied from zero (= idle) to one (= normalized maximum rate).

The model has the following parameters

Production:

α - rate of "dedication" to production, in fraction of time per unit of time; $\alpha \in [0, 1]$.

Hence the "accumulated commitment" of the manufacturing system to the piece is

$$\xi_t = \int_{\theta}^{t} \alpha_s \, ds$$

where θ is the instant of completion of the previously processed piece. Clearly the process (ξ_t), $t \geqslant 0$ satisfies $0 \leqslant \xi_t \leqslant \tau$ for all $t \geqslant 0$ and a completion time θ is characterized by discontinuity in the path of ξ_t.

Let θ_i $i = 1, 2, \ldots$ be the sequence of completion times. The counting process (q_t), $t \geqslant 0$ defined by

$$q_t = \sum_i I_{\{\theta_i \leqslant t\}}$$

is the accumulated production at each t.

Demand:

Request for these items arrives at random in time. The lot sizes are also random variables of discrete type. We assume that the requests are stationary in time and the arrival times form a Poisson process with rate λ. The lot sizes are i.i.d. with some given discrete distribution.

Let T_i, $i = 1, 2, \ldots$ be the sequence of request times and W_i, $i = 1, 2, \ldots$ be the correspondent lot sizes. The counting process (d_t), $t \geqslant 0$ defined below represents the accumulated demand at each time t:

$$d_t = \sum_i W_i \, I_{\{T_i \leqslant t\}}$$

Therefore, the storage process denoted (n_t), $t \geqslant 0$ is defined by

$$n_t = q_t - d_t , \quad t \geqslant 0$$

($n_t < 0$ denotes backlog).

Let (z_t), $t \geqslant 0$ be the state process defined by

$$z_t = (n_t , \xi_t) , \quad t \geqslant 0$$

$t \to z_t$ takes its values in the "infinite ladder" $Z \times [0, \tau]$ and a typical path is shown in the figure 1. This problem is an example of Piecewise Deterministic Markov Process for which the extended generator is given by

$$Af(z) = X^{\alpha} f(n, \xi) + \lambda \left[\sum_{k=1} f(n-k , \xi) p_k - f(n, \xi) \right]$$

where

χ^{α} = vector field parametrized by α

$\{p_k\}_{k=1,2,\ldots}$ = distribution of lot sizes : $p_k = P(W = k)$

For the characterization of Piecewise Deterministic Markov Process and examples see [1]. In fact the model we study here has a state space similar to the one of Capacity Expansion Planning to be found in that paper. In a way, these processes can be seen as an extension of the jump Markov disturbances model proposed in [4].

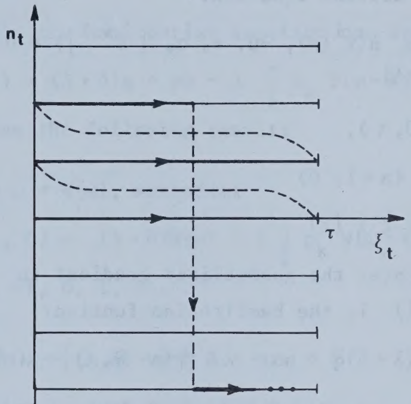


Figure 1 - A Typical path in the state space

THE CONTROL PROBLEM

Let c+ represents the cost of storage per item per unit time and let c^- represents shortage costs per item per unit time, associated with non-supplied demand. Let δ be a positive discount factor and represent

$n^+ = \max\{0, n\}$; $n^- = \max\{0, -n\}$

The stochastic control problem of itemized production to meet demand is defined as

$$\text{minimize } E\left[\int_0^{\infty} e^{-\delta t}\,(c^+ n_t^+ + c^- n_t^-)\,dt\right]$$

on the class of admissible policies. This problem has non-trivial solution (i.e. $\alpha_t \equiv 1$) if

$$E[W] < \frac{1}{\lambda\tau}$$

otherwise production resources cannot cope with demand.

We apply to this problem the control theory developed in [3]. One can check that the problem above verifies assumptions (A1) to (A5) fot the two main theorems of [3]. In this situation, theorems 1 and 2 establish the following:

Theorem 1 : Let $V^1 : Z \times [0,\tau] \to R_+$, be an arbitrary function which is Lipschitz continuous on the second component. A sequence $\{V^i\}$, $i = 1, 2, ..$ which is Lipschitz continuous on the second component of z tends uniformly to the value for the problem function, and the sequence satisfies the modified Hamilton-Jacobi-Bellman equation.

$$\min_{\mu \varepsilon \partial V^i(z)} \; - \; \max_{\alpha \varepsilon [0,1]} H(V^i(z), -\mu, z, \alpha, [V^{i-1}]) = 0$$

for each $z \;\varepsilon\; Z \times [0, \tau)$,

$$V^i(n, \tau) = V^{i-1}(n+1, 0)$$

for each $z = (n, \tau) \;\varepsilon\; Z \times \{\tau\}$

Here $\partial V(\cdot)$ indicates the generalized gradient in the sense of Clarke and $H(\cdot, \cdot, \cdot, \cdot, [\cdot])$ is the hamiltonian funtion:

$$H(q, p, z, \alpha, [\psi]) = (\lambda + \delta)q + p\alpha - \lambda E[\psi(n - \bar{W}, \xi)] - (c^+ n^+ + c^- n^-) \qquad (2.1)$$

Theorem 2 : The function $z \to V(z)$, the value for the problem, is Lipschitz continuous on the second component of z and it satisfies the H - J - B equation:

$$\min_{\mu \varepsilon \partial V(z)} \; - \; \max_{\alpha \varepsilon [0,1]} H(V(z), -\mu, z, \alpha, [V]) = 0$$

for each $z \;\varepsilon\; Z \times [0, \tau)$,

$$V(n, \tau) = V(n+1, 0)$$

for each $z = (n, \tau) \;\varepsilon\; Z \times \{\tau\}$.

Theorem 1 provides a sequential criterion of optimality; when functions V^i, $i = 1, 2, \ldots$ are differentiable it also provides a recursive procedure for solutions because the solution of H - J - B in this case is unique.

A CHANGE OF VARIABLE

Now, we take the real line and a family of simple functions $I_k(\omega)$, $k = \ldots, -1, 0, 1, \ldots$, $\omega \;\varepsilon\; R$ defined by

$$I_k(\omega) = \begin{cases} k & \text{if } \omega \in [k\tau, (k+1)\tau) \\ 0 & \text{otherwise} \end{cases} \qquad k = \dots, -1, 0, 1, \dots$$

And take a correspondence between $z \in Z \times [0, \tau]$ and $\omega \in R$ such that

$$z = (n, \xi) \qquad \omega = n\tau + \xi$$

We see that this correspondence is one-to-one.

Let $c(\omega)$ be defined by

$$c(\omega) = \begin{cases} c^+ I_k(\omega) & \text{if } \omega > 0 \\ c^- I_k(\omega) & \text{otherwise} \end{cases}$$

In view of (2.1) the hamiltorian function can be rewritten as

$$\tilde{H}(q, p, \omega, \alpha, [\psi]) = (\lambda + \delta)q + p\alpha - \lambda \sum_k p_k \psi(\omega - k\tau) - c(\omega)$$

and theorem 2 provides the following result:

Lemma 1. A function $\omega \to W(\omega)$, satisfies

$$\min \{\dot{W}(\omega), 0\} = (\lambda + \delta)W(\omega) - \lambda \sum_k p_k W(\omega - k\tau) - c(\omega)$$

or $\omega \notin k\tau \quad k = \dots, -1, 0, 1, \dots$
with

$$W(\omega) = V(z) \quad \text{for } z = (n, \xi) \text{ with } \omega = n\tau + \xi$$

and an optimal solution is such that $\alpha(\omega) = I_{\{\dot{W}(\omega) < 0\}}$.

Proof - Introducing the hamiltorian $\tilde{H}$ in theorem 2 and denoting $V(z) = W(\omega)$ for $\omega = n\tau + \xi$ when $z = (n, \xi)$, one obtains the expression in the lemma with the left hand side replaced by

$$\min_{\mu \in \partial W(\omega)} \min_{\alpha \in [0,1]} \mu.\alpha$$

if $\min\{\mu : \mu \in \partial M(\omega)\} \geqslant 0$ the above expression is null. If $\min\{\mu : \mu \in \partial W(\omega)\} < 0$ theorem 2 yields

$$\min \{\mu : \mu \in \partial W(\omega)\} = (\lambda + \delta)W(\omega) - \lambda \sum_k p_k W(\omega - k\tau) - c(\omega) \tag{3.1}$$

Since $\omega \to W(\omega)$ is (Lipschitz) continuous and $c(\omega)$ is constant on intervals $[k\tau, (k+1)\tau)$ $k = \dots, -1, 0, 1, \dots$ the expression above indicates that $\omega \to W(\omega)$ should be differentiable on $\omega \notin k\tau$, $k = \dots, -1, 0, 1, \dots$ in this situation $\partial W(\omega) = \dot{W}(\omega)$ on $\omega \notin k\tau$. The optimal solution proposed follows directly from (3.1) and considerations above.

THE ALGORITHM

Since $\omega \to c(\omega)$ is monotonically decreasing function for $\omega < 0$ and

$$W(\omega) = \inf_{\alpha} E^{\alpha}_{\omega} \left[\int_{0}^{\infty} e^{-\delta t} c(\omega_t)\, dt \right] \quad \text{where } \omega_t = n_t \tau + \xi_t$$

one has

$$W(\omega_1) \geqslant W(\omega_2) \quad \text{for } \omega_1 < \omega_2 < 0$$

and for $\omega_o < 0$ sufficiently small

$$(\lambda + \delta)\, W(\omega) - \lambda \sum_k p_k\, W(\omega - k\tau) - c(\omega) < 0$$

should hold for all $\omega \leqslant \omega_o$. Otherwise, from lemma 1 the expression above is equal to zero. But that corresponds to a policy of no production with shortage and that cannot be optimal for ω_o sufficiently small. Hence we can solve the following differential equation with multiple retards:

$$\dot{W}'(\omega) = (\lambda + \delta)\, W'(\omega) - \lambda \sum_k p_k\, W'(\omega - k\tau) - c(\omega) \tag{4.1}$$

and the value W should satisfy the equation above until $\dot{W}(\omega)$ changes signal. The problem with this approach is that boundary conditions are non-existent. If the equation above were stable, an effective approach would consist of guessing function $W(\omega)$ on an interval $[k\tau, (k+1)\tau)$ for k sufficiently small. Unfortunately that is not the case here (e.g. see Hale [2]).

Although a direct solution has failed we can resort to the theorem 1 for controlled PDP's and give a recursive form of solution:

Lemma 2 - Let $W^1 : \mathcal{R} \to \mathcal{R}_+$ be an arbitrary function which is continuous. Let a sequence $\{W^i\}$ be defined by

$$W^i(\omega) = W^{i-1}(\omega) \quad \text{on } \omega \in k\tau,\ k = \ldots, -1, 0, 1, \ldots \tag{4.2}$$

$$\dot{W}^i(\omega) = (\lambda + \delta) W^i(\omega) - \sum_k p_k\, W^{i-1}(\omega - k\tau) - c(\omega) \quad \text{on } \omega \notin k\tau,\ k = \ldots, -1, 0, 1, \ldots \tag{4.3}$$

whenever the right hand side is negative; otherwise define

$$W^i(\omega) = -\frac{1}{\lambda + \delta} \Big(\sum_k p_k\, W^{i-1}(\omega - k\tau) - c(\omega) \quad \text{on } \omega \notin k\tau,\ k = \ldots, -1, 0, 1, \ldots \tag{4.4}$$

It follows that $W^i(\omega) \to W(\omega)$ uniformly

We employ lemma 2 and the conclusions before the lemma to propose the following algorithm

STEP 1 - Take $W^i : \mathcal{R} \to \mathcal{R}_+$ arbitrary continous function

STEP 2 - Solve recursively (4.3) for all ω. The solution W' will also solve (4.2) and corresponds to the "never stop" policy

STEP 3 - $\dot{W}'(\omega) < 0$ for ω sufficiently small, so that $W'(\omega) = W(\omega)$ and

$\alpha(\omega) = 1$. As increases $\dot{W}(\omega)$ should change signal, say in ω_0. From ω_0 increasing solve simultaneously

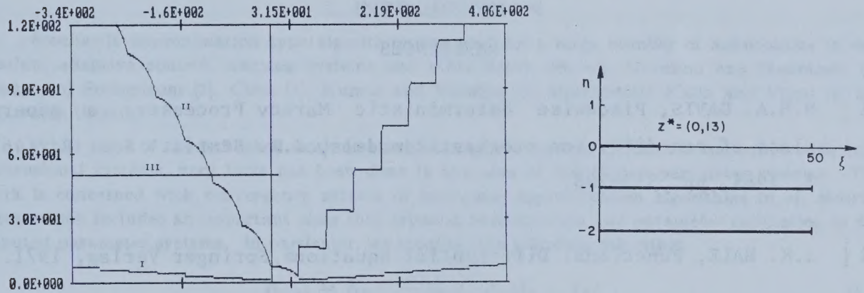
$$\dot{W}'(\omega) = (\lambda + \delta)\, W(\omega) - \sum_k p_k\, W(\omega - k\tau) - c(\omega)$$

and

$$W''(\omega) = -\frac{1}{\lambda + \delta}\left(\sum_k p_k\, W(\omega - k\tau) - c(\omega)\right)$$

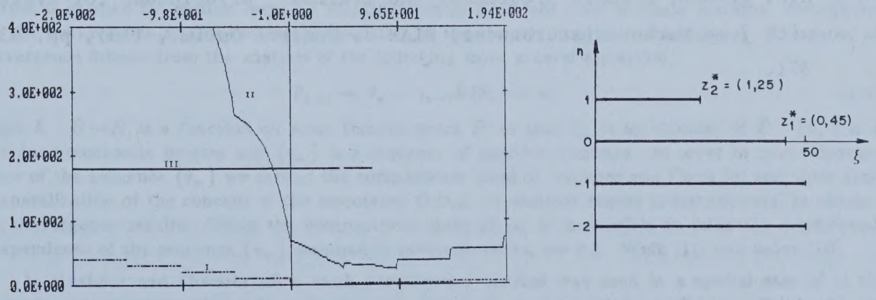
Take $W(\omega) = \min\{W'(\omega), W''(\omega)\}$ and $\alpha(\omega) = 1$ if $W'(\omega) < W''(\omega)$; $\alpha(\omega) = 0$ otherwise.

Figures 2, 3 and 4 bellow show solutions correspondent to three different cases of shortage costs. The parameter values are $\tau = 50$, $\delta = 0.05$, $\lambda = 0.02$ and $P(W = 1) = 1$ (i.e. demand in lots of size one).



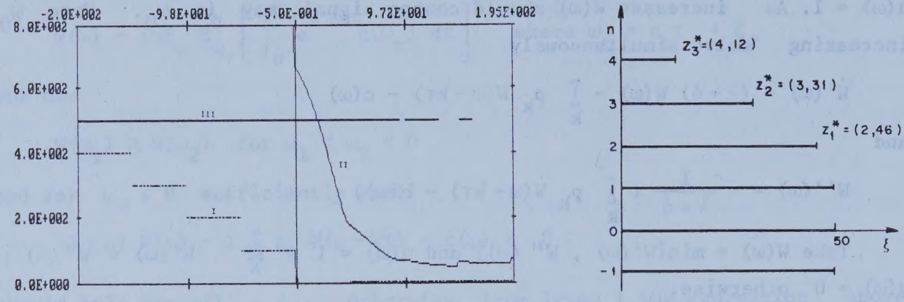
2(a) 2(b)

Fig 2(a) I. Cost Function $c^- = 1$, $c^+ = 1$; II. Value Function; III. Production Interval 2(b) Production locus in the state space z^* = decision point.



3(a) 3(b)

Fig 3(a) I. Cost Function $c^- = 10$, $c^+ = 1$; II. Value Function; III. Production Intervals 3(b) Production locus in the state space; z_1^*, z_2^* decision points.



4(a) 4(b)

Fig 4(a) I. Cost Function $c^- = 100$, $c^+ = 1$; II. Value Function; III. Production Intervals 4(b) Production locus in the state space; z_1^*, z_2^* and z_3^* decision points.

REFERENCES

[1] M.H.A. DAVIS, Piecewise deterministic Markov Processes: a general class of non-diffusion stochastic models, J.R. Statis. Soc. B, (46), 3, 1984, pp. 353-388.

[2] J.K. HALE, Functional Differential Equations, Springer Verlag, 1971.

[3] J.B. RIBEIRO DO VAL, M.H.A. DAVIS, Local and sequential criteria of optimality for control of piecewise deterministic Markov processes, to appear.

[4] R.W. RISHEL, Dynamic programing and minimum principles for systems with jump Markov disturbances, SIAM J. Control Optim., (13), pp. 338-371.

STOCHASTIC APPROXIMATION FOR DISTRIBUTED PARAMETER SYSTEMS

NADAV BERMAN   ADAM SHWARTZ
Aeronautical Engineering   Electrical Engineering

Technion - Israel Institute of Technology, Haifa, 32000, Israel

ABSTRACT

Several applications of the linear stochastic approximation algorithm

$$K_{n+1} = K_n - \frac{1}{n} X_n \circ [K_n X_n - Y_n]$$

to problems of identification of (possibly distributed) systems and optimization are given. The convergence of the linear algorithm is proved using the convergence with probability one of stochastic approximation algorithms of the form

$$\theta_{n+1} = \theta_n - \gamma_{n+1} h(\theta_n) + u_{n+1}$$

where the θ's belong to some Banach space and $\{u_n\}$ is a stochastic process. The applicability of conditions for convergence given here is demonstrated via an example. Convergence of the nonlinear algorithm is proved through an extension of the compactness ideas of Kushner and Clark [4].

I. INTRODUCTION

Stochastic approximation type algorithms are used for a large number of applications in estimation, adaptive control, learning systems and other fields; see e.g. Nevelson and Hasminski [9], Ljung and Soderstrom [7], Chen [1], Kumar and Varaiya [3], Herkenrath, Kalin and Vogel [2] and references therein.

While most of the published work on stochastic approximation methods is concerned with finite dimensional systems, very little has been done in the area of distributed parameter systems. This work is concerned with convergence aspects of stochastic approximation algorithms in an abstract setup which includes an important class that arises in identification and parameter estimation in distributed parameter systems. In particular, we consider the following algorithm

$$K_{n+1} = K_n - \frac{1}{n} X_n \circ [K_n X_n - Y_n] \tag{1.1}$$

where $\{X_n\}_n, \{Y_n\}$ are stochastic processes that take values in some Hilbert spaces H_1 and H_2 respectively. $K_n : H_1 \to H_2$ is a bounded linear operator, i.e. $K_n \in L(H_1, H_2)$ for all n, and $X \circ Y$, $X \in H_1$, $Y \in H_2$, denotes the tensor product between X and Y. We remark here that (1.1) includes the special case where stochastic approximation is applied to parameter estimation in a finite dimensional linear regression model where (1.1) is sometimes called the LMS algorithm.

Motivating the use of (1.1), we show how it naturally arises when abstract stochastic approximation is applied to impulse response identification in a certain class of linear systems. Conditions for the convergence of this recursive algorithm to the desired limit point are given. The proof of convergence follows from the analysis of the following more general algorithm,

$$\theta_{n+1} = \theta_n - \gamma_{n+1} h(\theta_n) + u_n \tag{1.2}$$

where $h : B \to B$ is a function on some Banach space B so that θ_n is an element of B, $\{u_n\}$ is a B-valued stochastic process and $\{\gamma_n\}$ is a sequence of positive numbers. In order to prove convergence of the sequence $\{\theta_n\}$ we extend the compactness ideas of Kushner and Clark [4] and show that a generalization of the concept of the associated O.D.E. to abstract spaces is instrumental in obtaining convergence results. Using the compactness ideas of [4] it is possible to relax the conditional independence of the sequence $\{u_n\}$ assumed in previous works, see e.g. Walk [11] and Salov [10].

In Kushner and Shwartz [5] a weak convergence method was used in a special case of (1.2), that allowed replacing the independence assumption by some mixing conditions. It should be pointed out that unlike the finite dimensional linear case, where O.D.E approach is not the only way to establish convergence, it is not clear how to extend other methods so that they can be utilized in establishing the type of convergence for the linear infinite dimensional case that is discussed in this work.

Notation: We shall be dealing with stochastic processes on some probability space $\{\Omega, \mathbb{F}, P\}$ (which we assume to be complete), and taking their values in Hilbert spaces H_i. For random variable $X \in H_1$ and $Y \in H_2$ we denote by $X \circ Y$ the tensor product between X and Y. Using the identification of H with H^*, we shall view $X \circ Y$ as an operator from H_1 to H_2, i.e. a member of $\mathbb{L}(H_1, H_2)$, or as a bounded linear functional on $H_1 \otimes H_2$, which we denote by $X \circ Y \in \mathbb{L}(H_1 \otimes H_2)$. On the space $\mathbb{L}(H_1, H_2)$ we shall put the topology which is compatible with the weak topology of $\mathbb{L}(H_1 \otimes H_2)$. Thus, with some abuse of standard terminology, a sequence $\{A_n\}$ in $\mathbb{L}(H_1, H_2)$ is said to converge in the *weak operator topology* to an operator A in $\mathbb{L}(H_1, H_2)$ if

$$((A_n - A)x, y)_{H_1} \to 0 \quad \text{for } all \quad x \in H_2 \text{ and } y \in H_1$$

We denote by $\mathbb{H}_w$ the space $\mathbb{L}(H_1, H_2)$ when equipped with this topology.
For a set A denote by $\bar{A}$ the closure of A and by ∂A its boundary. $co(A)$ denotes the convex hull of A, and $\overline{co}(A)$ its closure. We denote by E the mathematical expectation operator.

II. RESULTS AND APPLICATIONS

Motivation and the Linear Algorithm

Consider the following problem. Given is a record of data $\{X_n\}_n$, $\{Y_n\}_n$ considered to be sequences of random variables that take values in Hilbert spaces H_1 and H_2 respectively. Assume the X_n and Y_n are related by the following model

$$Y_n = KX_n + \psi_n \tag{2.1}$$

where $\{\psi_n\}_n$ (sometimes called an observation noise) is an H_2-valued sequence of r.v.'s which are independent of $\{X_n\}$, and $K : H_1 \to H_2$ is a bounded linear operator, i.e. $K \in \mathbb{L}(H_1, H_2)$. In the identification problem K is unknown and the problem is to find a "good" characterization of K from the record of available data. As in the finite dimensional case one may try to find K by the minimum variance criterion, i.e.

$$\min_{K \in \mathbb{L}(H_1, H_2)} \quad E(Y_n - KX_n, h)^2_{H_2} \qquad h \in H_2 \tag{2.2}$$

Standard assumptions on stationarity and boundedness of first and second moments lead to the following operator equation

$$KR = f \tag{2.3a}$$

where R and f are given by

$$R = E(X_n \circ X_n), \qquad f = E(X_n \circ Y_n) \tag{2.3b}$$

so that R is the unique symmetric operator in $\mathbb{L}(H_1)$ which is defined by

$$E(X_n, h_1)_{H_1}(X_n, h_2)_{H_1} = (Rh_1, h_2)_{H_1}, \qquad h_1,\ h_2 \in H_1,$$

and f is the unique operator in $\mathbb{L}(H_1, H_2)$ defined by

$$(fh_1, h_2)_{H_2} = E(X_n, h_1)_{H_1} \cdot (Y_n, h_2)_{H_2}, \qquad h_1 \in H_1,\ h_2 \in H_2.$$

Thus equation (2.3) can be written as

$$E\{K(X_n \circ X_n) - X_n \circ Y_n\} = 0 \tag{2.4}$$

where this is interpreted in the following sense

$$E\{(KX_n, h_2)_{H_2}(X_n, h_1)_{H_1} - (X_n, h_1)_{H_1}(Y_n, h_2)_{H_2}\} = 0$$

for all $h_1 \in H_1$ and $h_2 \in H_2$.

Motivated by the Robbins - Monro procedure as applied in the finite dimensional case the obvious infinite dimensional analog for the recursive estimation of K may be formulated as

$$K_{n+1} = K_n - \frac{1}{n}[K_n(X_n \circ X_n) - X_n \circ Y_n] \tag{2.5}$$

Note that from the definition of the tensor product it follows that $K_n(X_n \circ X_n) = X_n \circ (K_n X_n)$ so that (2.5) may be written in the more familiar form

$$K_{n+1} = K_n - \frac{1}{n} X_n \circ [K_n X_n - Y_n] \tag{2.6}$$

It will be convenient from the point of view of analysis to deal with the adjoint counterpart of (2.5) namely, we shall be considering the following algorithm

$$U_{n+1} = U_n - \frac{1}{n}[(X_n \circ X_n)U_n - Y_n \circ X_n] \triangleq U_n - \frac{1}{n}[R_n U_n - f_n^*] \tag{2.7}$$

Obviously, provided $U_o = K_o^*$ one has $U_n = K_n^*$ for all n. From this it is clear that convergence properties of (2.5) and (2.7) are equivalent. Note that in either case (2.5) or (2.7), the basic process is Hilbert valued, and the resulting stochastic approximations process K_n and U_n take values in a Banach space.

Examples

A large number of Engineering problems are characterized by models that are described by (2.2). We now give several examples that demonstrate the use of such models in an abstract setup. One specific example is image processing where an image is described by the (real valued) function $f(x,y)$. The image is processed by a system (which may be an optical channel, or an image processing system) whose effect is given by

$$g(x,y) = \int\int_{-\infty}^{\infty} h(x_1,y_1) f(x-x_1, y-y_1)\, dx_1 dy_1$$

where $g(x,y)$ describes the available picture, and $h(x,y)$ is the impulse response of the processing system. It is assumed that $\int\int_{-\infty}^{\infty} | h(x_1,y_1) |^2 dx_1 dy_1 < \infty$.

An algorithm to identify h, which uses samples of the picture along one axis (this may be the time axis) may be obtained as follows. One can write

$$g(x,n) = \int\int_{-\infty}^{\infty} h(x_1,y_1) f(x-x_1, n-y_1)\, dx_1 dy_1$$

Define X_n by $(X_n)(s,u) = f(s, n-u)$. Since f is a two parameter stochastic process then X_n may be considered an H_1-valued random variable where

$$H_1 = \{w(x,y): \mathbb{R}^2 \rightarrow \mathbb{R} : \int\int_{-\infty}^{\infty} | w(x,y) |^2 dx dy < \infty\}$$

Z_n is defined by $(Z_n)(x) = g(x,n)$, so that $Z_n = KX_n$ where

$$(Z_n)(x) = \int\int_{-\infty}^{\infty} h(x_1,y_1) f(x-x_1\,;\, n-y_1)\, dx_1 dy_1 \triangleq (KX_n)(x)$$

Thus $Z \in H_2$ where $H_2 = \{w: \mathbb{R} \rightarrow \mathbb{R}: \int_{-\infty}^{\infty} | w(x) |^2 dx < \infty\}$, under appropriate conditions. So, we can use stochastic approximations to estimate K from noisy observations, as in (2.1).

In general, stochastic approximations with linear dynamics arise when the underlying system is linear. Suppose, for example that $\{X(t)\}_t$ is a stochastic process that takes values in a Hilbert space H, and assume the input-output relation is given by

$$Z(t) = \int_0^1 h(\tau) X(t-\tau)\, d\tau \tag{2.8}$$

where $h(\tau) \in \mathbb{L}(H,H)$ for all $\tau \in [0,1]$. Assume $h(\tau)$ is strongly measurable (Lebesgue) on $[0,1]$ and that $\int_0^1 || h(\tau) ||_{HH}^2 d\tau < \infty$; a special case of (2.8) is a real linear system with finite memory. Suppose the observed output is obtained from sampled output with additive noise ψ_n, i.e.

$$\tilde{Y_n} = \int_0^1 h(\tau) X(n-\tau)\, d\tau + \psi_n \tag{2.9}$$

where $\{\psi_n\}_n$ are H-valued r.v.'s. Now define $(X_n)(\tau) \overset{\Delta}{=} X(n-\tau))$. Provided some assumptions (to be discussed later) on the processes involved are valid it is possible to view $\{Y_n\}$ $\{X_n\}$ $\{\psi_n\}$ as a discrete time stochastic process in an appropriate Hilbert space. In fact we may formulate these spaces as follows:

Let $H_1 = L_2([0,1],H)$, $H_2 = H$. Then $X_n \in H_1$, $Y_n \in H_2$, and $\psi_n \in H_2$. So one arrives at the model characterized by (2.1), where

$$Kf = \int_0^1 h(\tau) f(\tau)\, d\tau \quad , \quad \text{for all } f \in H_1$$

The integral is defined to be the Bochner integral. Thus, $K : H_1 \to H_2$ and it is bounded since by hypothesis $\int_0^1 \| h(\tau) \|_{H,H}^2 d\tau < \infty$. Note that in this case $K^* : H \to L_2([0,1],H)$ and is given by $(K^* g)(t) = h^*(t) g$. Thus the conjugate stochastic approximation algorithm is given by

$$h_{n+1}^*(t) = h_n^*(t) - \frac{1}{n}\left[\int_0^1 h_n(\tau) X(n-\tau)\, d\tau - Y_n\right] \circ X(n-t) \tag{2.10}$$

or, equivalently

$$U_{n+1}(t) = U_n(t) - \frac{1}{n} X(n-t) \circ \left[\int_0^1 U_n(\tau) X(n-\tau) d\tau - Y_n\right] \tag{2.11}$$

It is clear that $U_0(\tau) = h_0(\tau)$ on $[0,1]$ implies $U_n(t) = h_n(t)$ for all n. Note that in the case where $H = R$ (2.11) reduces to the algorithm considered by Kushner and Shwartz [5].

Another example is an approximation problem. Assume a system is described by

$$\frac{dU}{dt} = AU + X(t) \tag{2.12}$$

where A is the infinitesimal generator of a strongly continuous semigroup defined on a Hilbert space H. The solution is given by

$$U(t) = \int_0^t T(t-s) X(s) ds = \int_0^t T(s)(t-s) ds$$

(the initial conditions are assumed to be zero).

Assuming $\{X(t)\}$ is some stochastic process with values in H one concludes that $U(t)$ is a stochastic process in H. When $T(t)$ is stable, that is, $\| T(t)x \| \to 0$ for all $x \in H$, as $t \to \infty$, it makes sense in various instances to try and approximate $T(t)$ by an impulse response $h(t)$ that has a compact support, so that one may try to fit the data $\{X_n : (X_n)(t) = X(n-t)$ and $Y_n = U(n)\}$ to the model: $Y_n = \int_0^1 h(\tau) X(n-\tau) d\tau$ where $h(\tau)$ is an $\mathbb{L}(H)$ valued operator. Thus, once more one arrives at the model $Y_n = KX_n$.

Review of results

In the next section we prove convergence a.e. of the algorithm (1.1) in the weak operator topology, provided conditions L1-L6 stated below are satisfied. However, we state this convergence result in this section (Theorem 2.1 below) and give an example which demonstrates that these conditions are not vacuous. We use the following conditions:

L1. $\frac{1}{n}\sum_{k=1}^{n} X_k \circ X_k \to R$ a.e. as $n \to \infty$ in the uniform operator topology, where R is a symmetric trace class operator in $\mathbb{L}(H_1,H_1)$.

L2. $\frac{1}{n}\sum_{k=1}^{n} \| X_n \|_{H_1}^2 \to traceR$ a.e. as $n \to \infty$.

L3. $\frac{1}{n}\sum_{k=1}^{n} X_k \circ Y_k \to E(X_1 \circ Y_1) \overset{\Delta}{=} f$ a.e. in the uniform operator topology of $\mathbb{L}(H_1,H_2)$.

L4. $KR = f$ has a unique solution $K \in \mathbb{L}(H_1,H_2)$.

L5. There exists $\alpha > 0$ such that $\frac{1}{n}\sum_{k=1}^{n} k^{\alpha}(K(X_k \circ X_k) - X_k \circ Y_k) \to 0$ a.e. as $n \to \infty$ in the uniform operator topology in $\mathbb{L}(H_1, H_2)$.

L6. The range of R is dense in H_1.

Let H_1 and H_2 be separable Hilbert spaces and let $\{X_n\}$ ($\{Y_n\}$) be an H_1-valued (H_2-valued, resp.) stochastic process. Under these conditions we have

Theorem 2.1 : *Assume L1-L6. Then* $K_n \to K$ *with probability one in the weak operator topology, where* K *is the solution of (2.3a).*

Roughly speaking, R = covariance$(X_n) \overset{\Delta}{=} E(X_n \circ X_n)$ and $f = E(X_n \circ Y_n)$ (2.3b). This result is well known in the finite - dimensional case, under various boundedness and mixing conditions on the sequences $\{X_n\}$ and $\{Y_n\}$ [7].

The proof of this theorem is based on the extension of the well-known lemma of Kushner and Clark [4] from R^n to the abstract setting required here. The example below illustrates the validity of the conditions of Theorem 2.1.

Example: Let $\{X(t)\}_t$, $t \geq 0$ be a real valued zero mean, stochastic process defined on some probability space $(\Omega, \mathbb{F}, P)$. Assume $\{X(t)\}_t$ satisfies the following

B1. $\{X(t)\}_t$ is stationary (in the strict sense) and ergodic.

B2. $\frac{1}{T}\int_0^T X(t)dt \to EX(0) = 0$ P-a.e. as $T \to \infty$

B3. $\frac{1}{T}\int_0^T X(t)^2 dt \to E\,|X_o|^2 < \infty$ P-a.e. as $T \to \infty$.

B4. $X(t,\omega)$ is measurable as a function that takes $(R \times \Omega, \mathbb{B}(R) \times \mathbb{F})$ into $(R, \mathbb{B}(R))$.

B5. $\{X(t)\}$ is Gaussian.

Remark: under B5, H_1 can (and will) be chosen so that the null space of the covariance operator is the zero vector.

Suppose now that it is required to identify the impulse response $h(\tau)$ of the system whose output is given by

$$V(t) = \int_0^1 h(\tau)X(t-\tau)d\tau \tag{2.13}$$

where it is assumed that $\int_0^1 |h(\tau)|^2 d\tau < \infty$ so that h may be considered an element of $L_2[0,1]$. The observation of the output $V(t)$ is obtained by sampling the output $V(t)$. Taking the sampling interval to be one unit time, and assuming the observations are corrupted with noise one obtains an observation

$$Z_n \overset{\Delta}{=} V_n + \psi_n \overset{\Delta}{=} \int_0^1 h(\tau)X(n-\tau)d\tau + \psi_n \tag{2.14}$$

where $\{\psi_n\}_n$ is a sequence of i.i.d. r.v.'s (defined on $(\Omega, \mathbb{F}, P)$), $E(\psi_n) = 0$ $E(\psi_n^2) = \sigma^2$ for all n. It is also assumed that $\{\psi_n\}_n$ is independent of the entire history of $\{X(t)\}_t$, i.e. it is independent of the family $\{\mathbb{F}_t\}_t$ of sub σ-algebras generated by the process $\{X(t)\}_t$.

To cast the problem within the abstract setup that has been introduced before, let $H_1 = L_2[0,1]$, $H_2 = R$, and let X_n be an element of $L_2[0,1]$ defined by $(X_n)(t) = X(n-t)$, $t \in [0,1]$. Then X_n is an $L_2[0,1]$-valued random variable that is defined on $(\Omega, \mathbb{F}, P)$, and we can write

$$Z_n = (h, X_n) + \psi_n = KX_n + \psi_n \ , \tag{2.15}$$

where $KX_n = (h, X_n)_{H_1}$, namely K is simply an element of H_1 (since H_1 is real, H_1 is identified with H_1^*). The corresponding stochastic approximation algorithm is given by (cf. Kushner and Shwartz [5])

$$h_{n+1}(\tau) = h_n(\tau) - \frac{1}{n} X(n-\tau)\left[\int_0^1 h_n(\sigma) X(n-\sigma)\, d\sigma - Z_n\right], \quad \tau\in[0,1] \tag{2.16}$$

Utilizing now Theorem 2.1 and provided that assumptions L1-L6 are all satisfied, one concludes that the algorithm converges in the weak topology of $L_2[0,1]$ to the unique solution of the linear equation $KR = f$, where in this case R is characterized by

$$(Rg_1,g_2)_{H_1} = \int_0^1 \int_0^1 R(u-s) g_1(u) g_2(s)\, du\, ds, \quad g_1,g_2\in H_1 \; ; \quad R(u-s) = E[X(u)X(s)]$$

$$f = E(Z_n X_n) \; ; \quad (f)(t) = \left(\int_\Omega Z_n X_n \, dP\right)(t) = \int_\Omega Z_n X(n-t)\, dP$$

where the first integral is interpreted as a Bochner integral.

The lemmas stated and proved below provide a verification of the validity of L1 to L6, so that convergence a.s. in the topological weak sense follows from Theorem 2.1.

Lemma 2.2 *Under the hypotheses B1-B5 on $\{X(t)\}_t$, L1 is satisfied.*

Proof First recall that in this case $X_n \circ X_n$ is characterized by

$$((X_n \circ X_n) f_1, f_2)_{H_1} = (X_n, f_1)_{H_1}(X_n, f_2)_{H_1} = \int_0^1 X(n-s) f_1(s)\, ds \int_0^1 X(n-s) f_2(s)\, ds \quad f_1,f_2\in H_1$$

First we show that $\frac{1}{n}\sum_{k=1}^{n} X_k \circ X_k \to R$ a.e. in the uniform operator topology. To do this we make use of the ergodic theorem for Banach space-valued stochastic processes (see Parthasarathy [12, Theorem 9.4]). Let $\mathbb{H}$ be the subspace of $\mathbb{L}(H_1)$ that consists of the compact operators in $\mathbb{L}(H_1)$ equipped with the topology inherited by the uniform operator topology on $\mathbb{L}(H_1)$. With this topology $\mathbb{H}$ becomes a separable Banach space.

Define a function $g : H_1 \to \mathbb{H}$ by $g(h) \triangleq h \circ h$, $h\in H_1$. Obviously g is continuous. In fact one has $||g(h)||_{\mathbb{H}} = ||h||^2_{H_1}$. Let $\overline{X}_n(\omega) \triangleq g(X_n(\omega)) = X_n(\omega) \circ X_n(\omega)$. Then $\overline{X}_n : (\Omega,\mathbb{F}) \to (\mathbb{H},\mathbb{B})$ is strongly measurable, where $\mathbb{B}$ is the Borel σ-algebra generated by the strong topology on $\mathbb{H}$. It is easy to see that $\{\overline{X}_n\}_n$ is a stationary sequence of random variables that take values in $\mathbb{H}$. Let $(\mathbb{H}^\infty,\mathbb{B}^\infty)$ be the countable product of copies of the space $(\mathbb{H},\mathbb{B})$, and let μ be the measure induced by P on $(\mathbb{H}^\infty,\mathbb{B}^\infty)$. Define $\{Y_n\}_n$ to be the family of coordinate projections, i.e. if $h\in\mathbb{H}^\infty$ then $Y_n(h) = h_n$ where $h = (h_1, h_2 \cdots)$. Let T be the shift transformation on $\mathbb{H}^\infty$, that is $T(h_1,h_2,\ldots) = (h_2,h_3,\ldots)$. Then because of the stationarity of $\{\overline{X}_n\}_n$ μ is invariant under T. Also note that

$$\int_{\mathbb{H}^\infty} ||Y_1(x)||^2_{\mathbb{H}} d\mu(x) = \int_\Omega ||X_1||^2_{H_1} dP = E\int_0^1 [X(1-s)]^2 ds = E\,|X(0)|^2 < \infty \tag{2.17}$$

Let $\mathbb{F}_o$ denote the T-invariant sets in $\mathbb{B}^\infty$, i.e. $\mathbb{F}_o = \{A : A\in\mathbb{B}^\infty,\ T^{-1}(A) = A\}$. Recall now that $\mathbb{H}$ is separable so that the ergodic theorem for Banach space valued r.v.'s is applicable and one has $\lim_{n\to\infty} ||S_n - E(Y_1 | \mathbb{F}_o)||_{\mathbb{H}} = 0$ μ - *a.e.* where $S_n = \frac{1}{n}\sum_{k=1}^{n} Y_k$, and $E(Y_1|\mathbb{F}_o)$ is the conditional expectation in $\mathbb{H}$ (for definition see, e.g. Scalora [13]). This means that for P almost all $\omega\in\Omega$ the sequence

$$\{\overline{S}_n(\omega)\}_n \triangleq \{\frac{1}{n}\sum_{k=1}^{n} \overline{X}_n(\omega)\}_n = \{\frac{1}{n}\sum_{k=1}^{n} X_n(\omega) \circ X_n(\omega)\}_n$$

is a Cauchy sequence in $\mathbb{H}$, so that there exists a compact symmetric $\mathbb{L}(H_1)$-valued random variable $\overline{R}$ such that $||\overline{S}_n(\omega) - \overline{R}(\omega)||_{L(H_1)} \to 0$ as $n\to\infty$. We show now that $\overline{R}(\omega) = R$ $\quad P$ - a.e.

In order to do this it is sufficient to prove that $\{\overline{S}_n\}_n$ converges to R a.e. in the weak operator topology a.e., that is $(\overline{S}_n g_1, g_2)_{H_1} \to (Rg_1,g_2)_{H_1}$ for all g_1 and g_2 in H_1. Now,

$$(\bar{S}_n g_1,g_2)_{H_1} = ((\frac{1}{n}\sum_{k=1}^{n} X_k \circ X_k)g_1,g_2)_{H_1} = \frac{1}{n}\sum_{k=1}^{n}(X_k,g_1)_{H_1}(X_k,g_2)_{H_1} =$$

$$= \frac{1}{n}\sum_{k=1}^{n}\int_0^1\int_0^1 X(k-s)X(k-\sigma)g_1(s)g_2(\sigma)dsd\sigma = \int_0^1\int_0^1 [\frac{1}{n}\sum_{k=1}^{n} X(k-s)X(k-\sigma)]g_1(s)g_2(\sigma)dsd\sigma$$

We prove that the last term converges to $\int_0^1\int_0^1 R(s-\sigma)g_1(s)g_2(\sigma)dsd\sigma = (Rg_1,g_2)_{H_1}$. As $\{X(t)\}_t$ is, by hypothesis ergodic, we have

$$\frac{1}{n}\sum_{k=1}^{n} X(k-s)X(k-\sigma) \to R(s-\sigma) \tag{2.18}$$

for almost all (ω,s,σ) with respect to $[P\times(l\times l)]$ where $l\times l$ is the Lebesgue measure on the unit square. In order to show the required convergence it is sufficient to show that for those $\omega\in\Omega$ for which (2.18) holds the sequence of random variables $\{\xi_n(\sigma,s)\}_n$ defined by $\xi_n(\sigma,s) = \frac{1}{n}\sum_{k=1}^{n} X(k-s)X(k-\sigma)$ are $(l\times l)$ uniformly integrable. It is easy to show that

$$\sup_n \int\int_0^1 |\xi_n(\sigma,s)|^{1+\alpha} d\sigma ds < \infty \tag{2.19}$$

for some $\alpha > 0$. In fact choose $\alpha = 1$ so that one has

$$\int\int_0^1 |\xi_n(s,\sigma)|^2 d\sigma ds = \int\int_0^1 [\frac{1}{n}\sum_{k=1}^{n} X(k-s)X(k-\sigma)]^2 d\sigma ds \leq \int\int_0^1 \frac{1}{n}\sum_{k=1}^{n} X^2(k-s)\frac{1}{n}\sum_{k=1}^{n} X^2(k-\sigma)dsd\sigma$$

$$= [\frac{1}{n}\sum_{k=1}^{n}\int_0^1 X^2(k-s)ds]^2 = [\frac{1}{n}\int_0^n X^2(s)ds]^2 \to [E|X(0)|^2]^2$$

which is finite by assumption B3. This then implies (2.19), and the lemma follows.

Lemma 2.3 *Under B1-B5, L2 holds, that is,* $\frac{1}{n}\sum_{k=1}^{n} ||X_k||_{H_1}^2 \to$ *trace* $R \quad P$ *- a.e.*

Proof: $\frac{1}{n}\sum_{k=1}^{n} ||X_k||_{H_1}^2 = \frac{1}{n}\sum_{k=1}^{n}\int_0^1 X^2(k-s)ds = \frac{1}{n}\int_0^n X^2(s)ds \to E|X(0)|^2$ (by B3).

On the other hand, let $\{e_i\}_i$ be any ortho-normal basis for H_1. Then

$$traceR = \lim_{n\to\infty}\sum_{k=1}^{n}(Re_k,e_k)_{H_1} = \lim_{n\to\infty}\sum_{k=1}^{n} E(X_1,e_k)^2 = \lim_{n\to\infty} E\{\sum_{k=1}^{n}(X_1,e_k)^2\}.$$

Since $\sum_{k=1}^{n}(X_1,e_i)_{H_1}^2 \to ||X_1||_{H_1}^2$ monotonically as $n\to\infty$ it follows that $\sum_{k=1}^{n}(X_1,e_i)^2 \leq ||X_1||_{H_1}^2$. But $E||X_1||_{H_1}^2 < \infty$ so that Lebesgue dominated convergence theorem applies and we conclude that $traceR = E||X_1||_{H_1}^2$. Since $E||X_1||_{H_1}^2 = E\int_0^1 X(1-s)^2 ds$, Fubini's theorem now implies

$$E||X_1||_{H_1}^2 = \int_0^1 [EX^2(1-s)]ds = E|X(0)|^2$$

so that $traceR = E|X(0)|^2$. This completes the proof of the lemma.

Lemma 2.4 *Under B1-B5, L3 holds, that is,* $\frac{1}{n}\sum_{k=1}^{n} f_k \to f$ *a.e. in the strong topology of* H_1.

Proof Note that $f_n = Z_n X_n = X_n(X_n,h)_{H_1} + \psi_n X_n$, and $\frac{1}{n}\sum_{k=1}^{n} X_n(X_n,h)_{H_1} \to Rh$ by lemma (2.2). Thus it is left to show that $\frac{1}{n}\sum_{k=1}^{n}\psi_n X_n \to 0$ a.e. in the strong topology of H_1.

Let $Y_n = \psi_n X_n$. Obviously $\{Y_n\}_n$ is a stationary sequence of H_1-valued r.v.'s. So using the same chain of arguments as in the first lemma one arrives at the conclusion that $S_n \overset{\Delta}{=} \frac{1}{n}\sum_{k=1}^{n} Y_k$ is a Cauchy sequence in H_1 P - a.e.

Define now a sequence of real valued r.v.'s $\xi_n^y = \psi_n (X_n, y)_{H_1}$ $y \in H_1$.

For each $y \in H_1$ $\{\xi_n^y\}$ is obviously a martingale difference. Moreover

$$\sum_{n=1}^{\infty} \frac{E\,|\,\xi_n^y\,|^2}{n^2} = \sigma^2 \sum_{n=1}^{\infty} \frac{E\,(X_n, y)_{H_1}^2}{n^2} = \sigma^2 (Ry, y)_{H_1} \sum_{n=1}^{\infty} \frac{1}{n^2} < \infty .$$

Using now Chow's result ("stability theorem") it follows that $\frac{1}{n}\sum_{k=1}^{n} \xi_n^y \to 0$ $y \in H_1$ a.e.

Since H_1 is separable, it is not difficult to see that there exists $A \subset \Omega$ with $P(A) = 1$ such that $\frac{1}{n}\sum_{k=1}^{n} \xi_n^y \to 0$ on A for all $y \in H_1$. This completes the proof of the lemma.

Lemma 2.5 *Under B1-B5,* $Rh = f$ *has a unique solution* $h \in H_1$*, so that L4 holds.*
Proof: By definition (cf. 2.15), under L1-L3, $f \in \mathrm{Range}(R)$, hence there exists a solution $h \in H_1$. Since $\{X(t)\}_t$ is Gaussian the range of R is dense in H_1, and so the solution is unique.

Remark: Assumption B5 is used only in Lemma 2.5, and it is clear from the proof that under the weaker assumption that the range of R is dense the conclusions still hold.

Lemma 2.6 *Under B1-B5 there exists* $\alpha > 0$ *such that* $\frac{1}{n}\sum_{k=1}^{n} k^{\alpha}(R_k h - f_k) \to 0$ *a.e. in the strong topology of* H_1*, so that L5 holds.*

Proof Let $\alpha < \frac{1}{2}$ and define $\xi_n^y = n^{\alpha}\psi_n (X_n, y)_{H_1}$, $y \in H_1$. Observe now that $\sum_{n=1}^{\infty} \frac{E\,|\,\xi_n^y\,|^2}{n^2} \le \sigma^2 (Ry, y)_{H_1} \sum_{n=1}^{\infty} \frac{1}{n^{2-2\alpha}} < \infty$, and that ξ_n^y is a martingale difference. Therefore, exactly as in the proof of Lemma 2.4 one concludes that $\frac{1}{n}\sum_{k=1}^{n} \xi_n^y \to 0$, P - a.e. for all $y \in H_1$. the rest of the proof goes exactly as in the proof of Lemma 2.4.

III. EXTENSION OF THE KUSHNER - CLARK LEMMA TO BANACH SPACES

In this section we state convergence results for a deterministic recursive algorithm. Using the basic method of Kushner and Clark [4] we extend their result to the abstract case. The stochastic version of this result is then utilized in proving convergence a.s. (in some weak operator topology) of the linear algorithm. Proofs of results in this section are omitted. For detailed proofs, the interested reader is referred to Shwartz and Berman [14].

Let B be a Banach space. Fix $\theta_o \in B$ and define the B-valued sequence $\{\theta_n\}$ by

$$\theta_{n+1} = \theta_n - \gamma_{n+1} h(\theta_n) + u_{n+1} \tag{3.1}$$

where $u_i \in B$, $h : B \to B$ and the numbers γ_i satisfy the hypothesis:

$A\,1.$ $\gamma_i > 0$, $\sum_{i=0}^{n} \gamma_i \to \infty$, $\gamma_i \to 0$

In what follows we give conditions on the sequences $\{\theta_n\}$, $\{u_n\}$ and on the function h such that $\theta_n \to \theta^*$, and characterize θ^*. Interpolating $\{U_i\}$, $\{\theta_i\}$, we obtain B-valued functions in the following way; set $t_n = \sum_{i=0}^{n} \gamma_i$ and $m(n, T) = \max\{j : \sum_{i=n}^{j} \gamma_i \le T\}$ and define $\theta^o(t) = \theta_n$ on $[t_{n-1}, t_n)$, $\theta^o(t) = \theta_o$ for $t \le 0$ and $\overline{\theta}^n(t) = \theta^o(t + t_n)$.

Let $\theta^n(t)$ be the linear interpolation-version of $\overline{\theta}^n(t)$, i.e. $\theta^n(t) = \overline{\theta}^n(t)$ if t is a jump point of $\overline{\theta}^n(t)$, and $\theta^n(t)$ is linear between such points. Set $\overline{U}^o(t) = \sum_{i=0}^{n} u_i$ on $[t_{n-1}, t_n)$, $\overline{U}^o(t) = 0$ for $t < 0$. Let $U^o(t)$ be the linear interpolation as defined above, and $U^n(t) = U^o(t+t_n) - U^o(t_n)$.

Let $F \subset B$ be a convex set containing the origin, and assume it is compact in the topology induced by some invariant metric $r(x,y) \overset{\Delta}{=} r(x-y)$. Assume:

A2. $\theta^n(t) \in F$ for all n and t.

A3. $h : F \to F$ continuous and $co(h(F))$ (the convex hull of $h(F)$) is bounded.

A4. $\lim_{t \to \infty} \sup_{|s| \leq T} r(U^o(t+s) - U^o(t)) = 0$ for each T. Moreover, for each T, $\theta^n(t) - U^n(t) \in F$ for all $|t| \leq T$, provided n is large enough.

A5. For all F-valued cadlag (right continuous with left hand limits) functions η, $\eta_n \to \eta$ implies $\int_o^t \eta_n(s)ds \to \int_o^t \eta(s)ds$, where convergence is assumed relative to the topology induced by the metric r and uniform on bounded intervals.

Remark: The hypotheses above are exactly those used in the finite dimensional case, except that the continuity of h has been relaxed in some results on w.p.1 and on weak convergence; see e.g. Kushner [15], Benveniste, Goursat and Ruget [17], Kushner and Shwartz [16].

Consider now the differential equation

$$\frac{d\,\theta(t)}{dt} = -\,h(\theta(t)), \qquad t \in (-\infty, \infty) \tag{3.2}$$

A6. θ^* is an asymptotically stable solution of (3.2) in D, a domain of attraction for θ^* (assumed to be open), and all solutions of (3.2) with $\theta(o,x) \in D$ are continuous in x; where $\theta(t,x)$ is a solution of (3.2) with $\theta(o,x) = \theta(o) = x$.

Theorem 3.1: *Let $K \subset D \cap F$ be compact, and assume A1 - A6 hold. Then $\theta_n \in K$ infinitely often implies $\theta_n \to \theta^*$ in the topology induced by the metric r.*

Convergence of the linear algorithm

As mentioned above, the proof of Theorem 2.1 is a consequence of the stochastic version of Theorem 3.1. In fact as in the finite dimensional analog we can write (2.7) as follows

$$U_{n+1} = U_n - \frac{1}{n}[RU_n - f^*] + \frac{1}{n}(R - R_n)U_n + \frac{1}{n}(f_n^* - f^*) \tag{3.3}$$

so that in the terminology of (3.1) one has

$$h(\theta) = R\,\theta - f^*, \quad u_{n+1} = \frac{1}{n}(R - R_n)U_n + \frac{1}{n}(f_n^* - f^*),$$

and the associated O.D.E. becomes:

$$\frac{d\,\theta(t)}{dt} = -R\,\theta + f^*. \tag{3.4}$$

In the finite dimensional case, it is possible to prove convergence of (2.7) and (3.3) directly with no reference to the O.D.E., (see e.g. Ljung [6]), but the assumption on strict positivity of R is essential. Since this assumption is never valid in the infinite dimensional case for compact R, the use of the O.D.E. approach is essential.

The proof of Theorem 2.1 goes roughly as follows. One shows that under L1-L6 $\{U_n\} \subset F$ a.s., where F is a bounded set in $\mathbb{L}(H_1, H_2)$. The boundedness of F implies compactness of F in the weak operator topology. Thus F is metrizable with a metric that induces the same topology on F as the one inherited by the weak operator topology. One then shows that L1-L6 imply A1-A6, so that Theorem 3.1 is applicable for almost all $\omega \in \Omega$, i.e.

$$((U_n - U)h_2, h_1)_{H_1} \to 0 \quad a.s. \quad \text{for all } h_1 \in H_1,\ h_2 \in H_2 .$$

This obviously implies the conclusion of Theorem 2.1.

REFERENCES

[1] Chen, H. -F., *Recursive estimation and control for stochastic systems,* Wiley, New York, 1985.

[2] Herkenrath, U., D. Kalin, and W. Vogel, *Mathematical learning models - theory and applications,* Springer-Verlag, New York, 1983.

[3] Kumar, P. R. and P. Varaiya, *Stochastic systems; estimation, identification and adaptive control,* Prentice-Hall, New Jersey, 1986.

[4] Kushner, H. J. and D. S. Clark, *Stochastic Approximation methods for constrained and unconstrained systems,* Springer Verlag, 1978.

[5] Kushner, H. J. and A. Shwartz, "Stochastic Approximations in Hilbert space: identification and optimization of linear continuous-parameter systems," *SIAM J. Control Opt.*, vol. 23, no. 2, pp. 774-793, Sept 1985.

[6] Ljung, L., "Analysis of stochastic gradient algorithms for linear regression problems," *IEEE Trans. Info. Theory,* vol. IT-30, no. 2 Part I, pp. 151-160, March 1984.

[7] Ljung, L. and T. Soderstrom, *Theory and practice of recursive identification,* MIT Press, Cambridge, Mass., 1983.

[8] Metivier, M. and P. Priouret, "Applications of a Kushner and Clark Lemma to general classes of stochastic algorithms," *IEEE Trans. Info. Theory,* vol. IT 30, pp. 140-150, March 1984.

[9] Nevel'son, M. B. and R. Z. Has'minskii, *Stochastic Approximations and Recursive Estimation,* Translation of Mathematical Monographs Vol. 47, AMS, Providence, Rhode Island, 1973.

[10] Salov, G. I., "Stochastic approximation theorem in a Hilbert space and its application," *Theor. Prob. Appl.*, vol. 24, pp. 413-419, 1979.

[11] Walk, H., "An invariance principle for the Robbins Monro process in a Hilbert space," *Z. Wahr. verw. Gebiete,* vol. 39, pp. 135-150, 1977.

[12] Parthasarathy, K. R., *Probability measures on metric spaces,* Academic press, 1967.

[13] Scalora, F. S., "Abstract Martingale convergence theorems," *Pacific Journal of Math.*, vol. 11, pp. 347-374, 1961.

[14] Shwartz, A. and N. Berman, "Abstract Stochastic Approximation and Applications", EE Pub. No. 630, Israel Inst. of Tech., June 1987.

[15] Kushner, H. J., *Approximation and Weak Convergence Methods for Random Processes, with Applications to Stochastic Systems Theory,* MIT Press, Cambridge, Massachusetts, 1984.

[16] Kushner, H. J. and A. Shwartz, "An invariant measure approach to the convergence of Stochastic Approximations with state-dependent noise," *SIAM J. Control Opt.*, vol. 22, no. 1, pp. 13-27, January 1984.

[17] Benveniste, Albert, Maurice Goursat, and Gabriel Ruget, "Analysis of stochastic approximation schemes with discontinuous and dependent forcing terms with applications to data communication algorithms," *IEEE Auto. Control,* vol. AC-25, no. 6, pp. 1042-1058, December 1980.

[18] Kushner, H. J. and A. Shwartz, "Convergence of Stochastic Approximations with state- dependent noise under weak conditions," *Proc. 21st IEEE Conference on Decision and Control,* vol. 2, pp. 517-521, December 1982.

EDGE DETECTION BY COST MINIMIZATION

H. L. TAN, E. J. DELP, and S. B. GELFAND.
Purdue University
School of Electrical Engineering
Computer Vision and Image Processing Laboratory
West Lafayette, Indiana 47907.

ABSTRACT

Edge detection is an important task in image processing. Information from local and global edge configuration can be exploited in the detection process. In this paper, we present an algorithm in which spatial pointwise decisions on the presence of edges are made by considering information from the image data and the surrounding edge structure. Using this information, we are able to reduce fragmentation and produce edges that are thin and well localized. The detection process is cast as a problem in cost minimization and we use a heuristic search procedure to find low cost edge configurations.

1. INTRODUCTION

The detection of edges in an image is an important task in image processing. Its importance cannot be over-emphasized as it is often the first processing stage in object recognition and image understanding systems [1]; the accuracy in which edge detection can be performed is a crucial factor in determining the overall system performance.

The majority of edge detectors emphasize the use of difference operators. A major difficulty with such operators is that they are not robust with respect to noise, and the end result of applying them to real images usually show a high degree of false and fragmented edges. A significant amount of research has been devoted to the removal of noise effects and to the finding of optimal edge boundaries, for instance, through a process of smoothing or edge linking [2, 3]. Canny [4] approached the problem of detecting edges by designing one dimensional optimal filters that satisfy a set of performance criteria. The optimal filter for step edges was approximated by the derivative of a Gaussian function. Hildreth [5] used a rotationally invariant second derivative operator, the Laplacian, on Gaussian smoothed images. In contrast to using derivative operators, surface fitting techniques [6, 7]. can also be used to find changes in the image intensity profile.

In this paper, we describe an edge detection algorithm that finds edges by minimizing a cost function. Using this approach, we are able to exploit information from local edge configuration in the detection process; spatial pointwise decisions on the presence of edge pixels are made by considering information from local edge structure about those pixels. By using this information we are able to produce better edges in terms of accuracy in position, and edge continuity. As an example, consider the case of a fragmented edge that is the result of noise; a detection algorithm that exploits information from local edge configuration will be able to use this information to reduce the amount of fragmentation by linking locally disconnected edge fragments. By considering local edge structure, we are in fact exploiting information from neighboring pixels based on past decisions at these pixels. Attempts to formulate the problem of edge detection using edge configuration information in an optimal manner usually result in problems that are NP-complete (non-deterministic polynomial time complete). The search space for the optimal solution becomes prohibitively large as the number of possible solutions is equal to 2^N, where N is the number of pixels in the image.

In Section 2, we specify what we mean by an edge and describe some desirable characteristics of an edge. In Section 3, we define a comparative cost function which we will use to evaluate the relative quality of pairs of edge configurations. In

Section 4, we describe a heuristic search procedure to find low cost edge configurations for a given image. Experimental results of the application of the above algorithm to test images and natural scenes are given in Section 5.

2. CONCEPT OF AN EDGE

In this section, we specify what we mean by an edge, and establish common ground for discussing the following edge detection algorithm. No absolute definition of an edge exists and edge detection algorithms are only as good as their inherent assumption of what edges are. We define an edge to be a boundary in an image that separates two adjoining regions that have significantly dissimilar characteristics. The dissimilarity may be due to geometry, illumination, surface characteristics or orientation of the physical environment. The measure of region dissimilarity may be appropriately defined for various types of edges. For example, to find intensity edges such as step or ramp edges, we could define the measure to be the difference of gray level averages on either sides of the edge. Higher order statistical measures could be used, in which case, we would be finding edges separating regions of different texture. The above definition of an edge encompasses many different kinds of edges as long as a suitable measure of region dissimilarity can be specified for the edge of interest.

In addition to partitioning dissimilar regions, edges should exhibit the following characteristics:

1) Thin
Edges are boundaries, which by definition are narrow. Ideally, they should be only one pixel wide.

2) Accurately localized
It is desirable that the detected edge should lie in a position which maximizes the measure of dissimilarity between the two dissimilar regions. We specify that an edge is accurately localized when it achieves the above criterion.

3) Minimally fragmented
Fragmentation is the term given to edges that are sporadically discontinuous. This effect is dominant in algorithms that are overly sensitive to noise in the image. Correct edges should, as far as possible, exhibit a continuity that reflects the nature of the boundary in the physical environment.

4) At least 3 pixels long
Noise and fine texture may cause the appearance of short scattered "edges" of one or two pixels in length. We will omit such short "edges" and restrict our concept of edges to those that are greater than or equal to three pixels in length.

Practically, there is often a tradeoff between the different desirable characteristics of an edge. Due to conflicting edge requirements, situations arise where one is not able to simultaneously achieve two or more characteristics. It is thus appropriate to associate a measure of importance with each of these characteristics to resolve such incompatibilities.

3. COMPARATIVE COST FUNCTION

The objective of the detection algorithm is to find the pixels in an image that satisfy the above criterion of an edge. We do this by casting edge detection as a problem in cost minimization. Due to computation complexity, we are unable at this point to formulate an absolute cost function for finding edges; instead, we use a comparative cost function. The approach used to find a good edge configuration is to recursively generate new configurations that are compared to the previous best by means of the comparative cost function. The cost function evaluates the relative quality of pairs of edge configurations by considering their edge structure and examining how well the edges fit the image data. It should mathematically capture the intuitive concept of an edge which we have previously discussed. This section describes the formulation of the comparative cost function.

An image is a rectangular array of pixels g(m,n), $1 \leq m \leq m_{max}$, $1 \leq n \leq n_{max}$, where each pixel value g(m,n) is the gray level at location (m,n) in the image. Similarly, we define an edge configuration as a rectangular array of pixels p(m,n), $1 \leq m \leq m_{max}$, $1 \leq n \leq n_{max}$, where each pixel value p(m,n) takes on a binary value 0 or 1. If p(m,n) = 1, the pixel at location (m,n) is an edge pixel; otherwise it is a non-edge pixel.

3.1. Defining the Comparative Cost Function

Given a pair of candidate edge configurations s_i and s_j that differ only at one pixel location l =(m,n) , we define the comparative cost function $\Delta C(s_i,s_j)$ as :

$$\Delta C(s_i,s_j) = \sum_{k=1}^{k=M} w_k \left[C_k(s_j,l) - C_k(s_i,l) \right] \quad ; \quad w_k \geq 0 \quad \text{and} \quad 0 \leq C_k \leq 1 \tag{1}$$

$$= \sum_{k=1}^{k=M} w_k \Delta C_k(s_i,s_j) \tag{2}$$

The function is a weighted sum of M cost factors, where each of the weight values are given by w_k. It should be noted that l is any location within the rectangular array of pixels. For ease of notation, we will write $\Delta C(s_i,s_j)$ as ΔC_{ij} or, when no confusion occurs, simply as ΔC. Also, we shall refer to the pixel at location l, simply as pixel l. Now, we specify that

$$\begin{aligned} \Delta C_{ij} &> 0 \quad => \quad s_i \text{ is a better configuration} \\ &< 0 \quad => \quad s_j \text{ is a better configuration.} \end{aligned}$$

The motivation for making ΔC_{ij} a weighted sum of cost factors C_k, is for each of the factors to capture, in some way, a desirable characteristic of edges. Ideally, each cost factor should affect one and only one characteristic so that the relative importance of each can be appropriately emphasized by their corresponding weights w_k. Practically, this is difficult to achieve as the different characteristics exhibit some form of dependency on each other. For instance, minimizing fragmentation implies producing edges that are more continuous, and hence, there will be less short scattered edges.

Before defining the cost factors, we first specify what we mean by valid edge structures. A valid 2-neighbor edge structure is a collection of 3 edge pixels which are arranged such that the resulting edge structure is continuous and does not turn by more than 45 degrees. It is termed a 2-neighbor structure since the center pixel has 2 neighboring edge pixels. Figures 1(a) - (d) show 4 valid 2-neighbor edge structures. Figure 1(e) is an invalid edge structure since the edge makes a 90 degree turn to the right. Taking into account rotations of the examples in Figure 1, there are 12 possible valid 2-neighbor edge structures. A valid 3-neighbor edge structure is one of the 8 possible structures shown in Figure 2.

We define five cost factors (M=5) for capturing the desirable characteristics of an edge. The rectangular grid of an edge configuration is visualized as an overlay on the image; the cost factors are computed by examining the local structure of the edge configuration about pixel l, and the underlying image data. In the following sections, we define the values of each of the cost factors C_k, which we will use to compute the value of ΔC_{ij} in Equation (1).

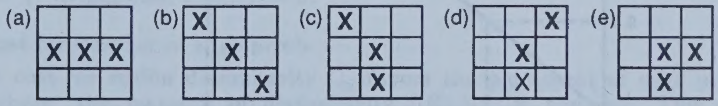


Figure 1. (a) to (d) are valid 2-neighbor edge structures,
(e) is an invalid edge structure.

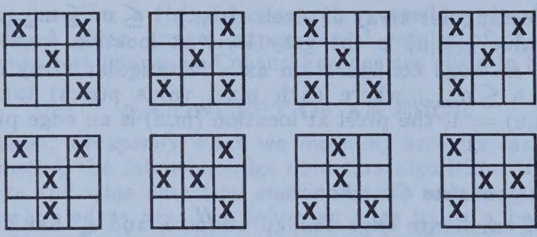


Figure 2. The 8 valid 3-neighbor edge structures.

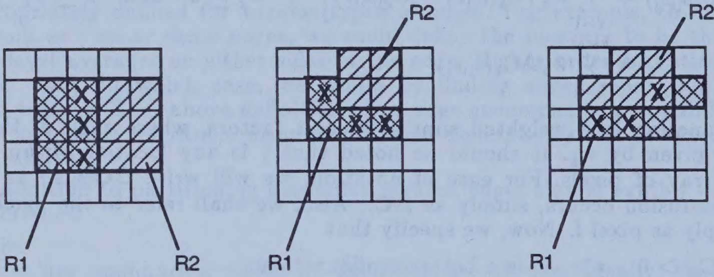


Figure 3. The regions R1 and R2 on either sides of an edge.

1) C_d: Cost for region dissimilarity.

For each valid 2-neighbor edge configuration, we define two regions R1 and R2, on either sides of the edge. Examples of these are shown in Figure 3 for three different edge structures. These regions have to be consistently defined so that edge positions are not ambiguous. The cost for region dissimilarity is based on a function f(R1,R2), that measures how different region R1 is from R2. The measure could be a simple difference of gray level averages in R1 and R2, or it could be more complicated measures based on statistical or structural properties in the gray levels. For our implementation, we first compute the magnitude d, of the difference of gray level averages in R1 and R2.

$$d = \left| \frac{1}{|R1|} \sum_{(i,j)\in R1} g(i,j) - \frac{1}{|R2|} \sum_{(i,j)\in R2} g(i,j) \right|$$

where $|R1|$ denotes the number of pixels in R1.

This difference is then mapped onto the unit interval [0,1] using the piecewise linear function m(d), shown in Figure 4. We see that f(R1,R2) = m(d) is dependent on the

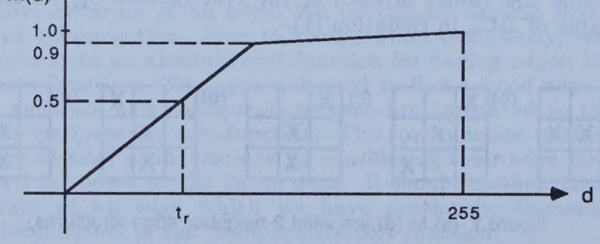


Figure 4. Piecewise linear function for computing C_d.

image data in the defined regions R1 and R2, and the parameter t_r, which we call the threshold. $C_d(s_i,l)$ is computed by examining the edge structure of s_i in a local 3×3 window neighborhood centered at pixel l. To determine the regions R1 and R2, we first have to decide which is the most appropriate valid 2-neighbor edge structure that would fit the window. This is done in the following way; there are 3 possible cases for the edge structure about pixel l:

Case 1: There are exactly 2 neighboring edge pixels which, together with an edge pixel at location l, form a valid 2-neighbor edge structure. This structure becomes the most appropriate edge structure.

Case 2: There are more than 2 neighboring edge pixels, one or more pairs of which, together with an edge pixel at location l, form possible valid 2-neighbor edge structures. Amongst these possible valid edge structures, the one which maximizes the measure of region dissimilarity is chosen as the most appropriate edge structure.

Case 3: If the local edge structure does not satisfy cases 1 or 2 above, then amongst the 12 possible valid 2-neighbor edge structures, the one which maximizes the measure of region dissimilarity at pixel l is chosen as the most appropriate edge structure.

If the pixel at l is an edge pixel, we set $C_d(s_i,l) = 0$. Otherwise, we use the following procedure to determine the value of $C_d(s_i,l)$. The most appropriate edge structure is chosen according to the rules above and $f(R1,R2,t_r) = m(d)$ is computed. We perform a non-maxima suppression by shifting the edge location in a direction determined by the chosen edge structure. For straight vertical, horizontal and diagonal edges, the shifting is performed by moving one pixel location in each of the opposite directions perpendicular to the edge. For all other edge structures, the shifting is done by moving the edge in each of the the four directions: up, down, left and right. If no larger measure of region dissimilarity results from shifting the edge structure, we set $C_d(s_i,l) = f(R1,R2,t_r)$, otherwise we set $C_d(s_i,l) = 0$;

2) C_t: Cost for edge thickness.

Using an 8-neighbor representation of the edge, we define a thick edge to be an edge structure that has multiple links between 2 or more of its edge points. A thin edge is an edge that is not thick. A thick edge pixel is defined as an edge pixel whose presence causes multiple links between its neighboring pixels. The cost for edge thickness is determined by considering pixel l in edge configuration s_i. If it is a thick edge pixel, then $C_t(s_i,l)$ is equal to one; otherwise it is equal to zero.

3) C_c: Cost for edge continuity.

This cost factor reduces fragmentation by assigning a cost value of 1 to single missing edge pixels that result in a disconnected edge. It examines configuration s_i in a local 5×5 window centered at pixel l. If pixel l is an edge pixel, we set $C_c(s_i,l) = 0$. Otherwise, if there are 2 short edges that could be connected to form a thin edge of at least 4 pixels long by placing an edge at pixel l, then $C_c(s_i,l) = 1$. For all other cases, the cost is zero.

4) C_l: Cost for edge length.

This assigns a cost of 1 to short edge pixels that are less than 3 pixels long. If pixel l in edge configuration s_i is part of an edge that is less than 3 pixels long, $C_l(s_i,l) = 1$. Otherwise, $C_l(s_i,l) = 0$.

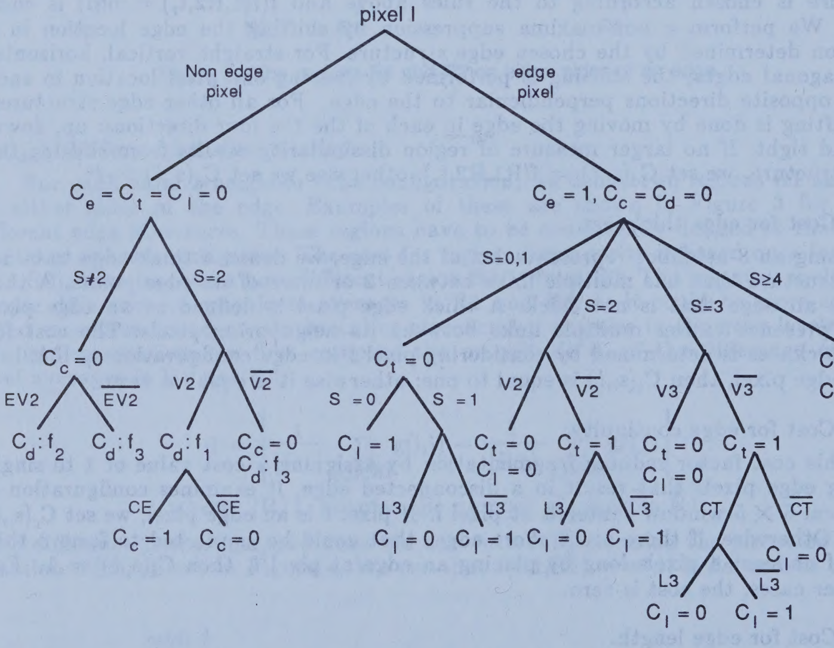
5) C_e: Cost for number of edge pixels.

The cost for region dissimilarity C_d favors the placement of edge pixels at all points where the measure of dissimilarity $f(R1,R2)$ is non-zero. This causes an excessive number of edges to be detected, and to suppress this, we assign a cost for each additional edge pixel detected. If pixel l is an edge pixel, then $C_e(s_i,l) = 1$; otherwise $C_e(s_i,l) = 0$.

The weights w_k associated with each cost factor are chosen according to the amount of emphasis that one chooses to place on each factor. The particular measure of dissimilarity f(R1,R2) used for the cost factor C_d has to be taken into account when selecting the weights.

3.2. Computing the cost

One approach to determine ΔC_{ij} would be to compute each cost factor independently and sum the difference as specified in Equation (1). However, a great deal of reduction in computation time can be achieved by pooling together information affecting each of the different factors and organizing it in a form that would allow for efficient computation. This is achieved by a decision tree structure as shown in Figure 5. The structure allows for the simultaneous computation of several cost factors by traversing the tree from root to leaf following the relevant path. A further reduction in computation time (by approximately half) is achieved by observing that configurations s_i and s_j differ only at pixel location l and ΔC_{ij} can be determined simply by considering s_i. We need not compute $C_k(s_i,l)$ and subtract it from $C_k(s_j,l)$; we can compute $\Delta C_k(s_{,}s_j)$ directly by considering s_i in the neighbor-


pixel l
Non edge pixel
edge pixel
$C_e = C_t = C_l = 0$
$C_e = 1, C_c = C_d = 0$
S≠2
S=2
S=0,1
S=2
S=3
S≥4
$C_c = 0$
V2
$\overline{V2}$
$C_t = 0$
V2
$\overline{V2}$
V3
$\overline{V3}$
$C_t = 1$
$C_l = 0$
EV2
$\overline{EV2}$
S =0
S =1
$C_d : f_2$
$C_d : f_3$
$C_d : f_1$
$C_c = 0$
$C_d : f_3$
$C_l = 1$
$C_t = 0$
$C_l = 0$
$C_t = 1$
$C_t = 0$
$C_l = 0$
$C_t = 1$
CE
$\overline{CE}$
L3
$\overline{L3}$
L3
$\overline{L3}$
CT
$\overline{CT}$
$C_c = 1$
$C_c = 0$
$C_l = 0$
$C_l = 1$
$C_l = 0$
$C_l = 1$
$C_l = 0$
L3
$\overline{L3}$
$C_l = 0$
$C_l = 1$

S = total number of edge pixels in 3 x 3 neighborhood

f_n : f(R1, R2) computed using case n

V2 : A valid 2-neighbor edge structure

V3 : A valid 3-neighbor edge structure

EV2 : Some pairs of neigboring edge pixels form a valid 2-neighbor edge structure

CE : Edge pixel at l will form a thin continuous edge of at least 4 pixels

L3 : Pixel l is part of a thin edge of at least 3 pixels

CT : Neighboring pixels are not clustered together in one corner or border of a 3 x 3 window

Figure 5. Decision tree for computing $C_k(s_i, l)$

hood of pixel l. The decision tree for this is shown in Figure 6. ΔC_{ij} is determined by appropriately traversing the tree from root to leaf following the relevant path. The tree assumes that configuration s_i does not have an edge at pixel l while s_j does. If the opposite is true, then ΔC_{ij} is determined simply by negating the value obtained from this tree.

4. A HEURISTIC SEARCH ALGORITHM

In this section, we discuss how the comparative cost function can be used to detect edges in an image. Particularly, we describe a heuristic search algorithm that finds low cost edge configurations.

The algorithm is as follows.

(1) Begin by selecting any arbitrary edge configuration s_i and any $l = (m,n)$, where $1 \leq m \leq m_{max}$ and $1 \leq n \leq n_{max}$.

(2) Select another edge configuration s_j such that it is identical to s_i except at pixel l, where it is the complement.

(3) Compute ΔC_{ij} and select the better of the two configurations. Call this configuration s_i.

(4) Pick a new value of $l = (m,n)$, where $1 \leq m \leq m_{max}$ and $1 \leq n \leq n_{max}$.

(5) If stopping criterion is not satisfied, Repeat from step (2).

The algorithm terminates when no better configuration can be found, or when a suitable stopping criterion is satisfied. A simple stopping criterion is the number of better configurations found after K iterations. If the number does not exceed a certain minimum, the algorithm stops. The new value of l may be selected either in a deterministic or random manner; comparisons have been made and practically it has been found to have little effect on the final output. However, it is essential that every possible pixel location be selected at least once. Typically, when l is chosen by sequentially stepping through the image at each pixel location, 3 to 5 iterations through the image is sufficient for the algorithm to converge to the stopping criterion.

5. EXPERIMENTAL RESULTS

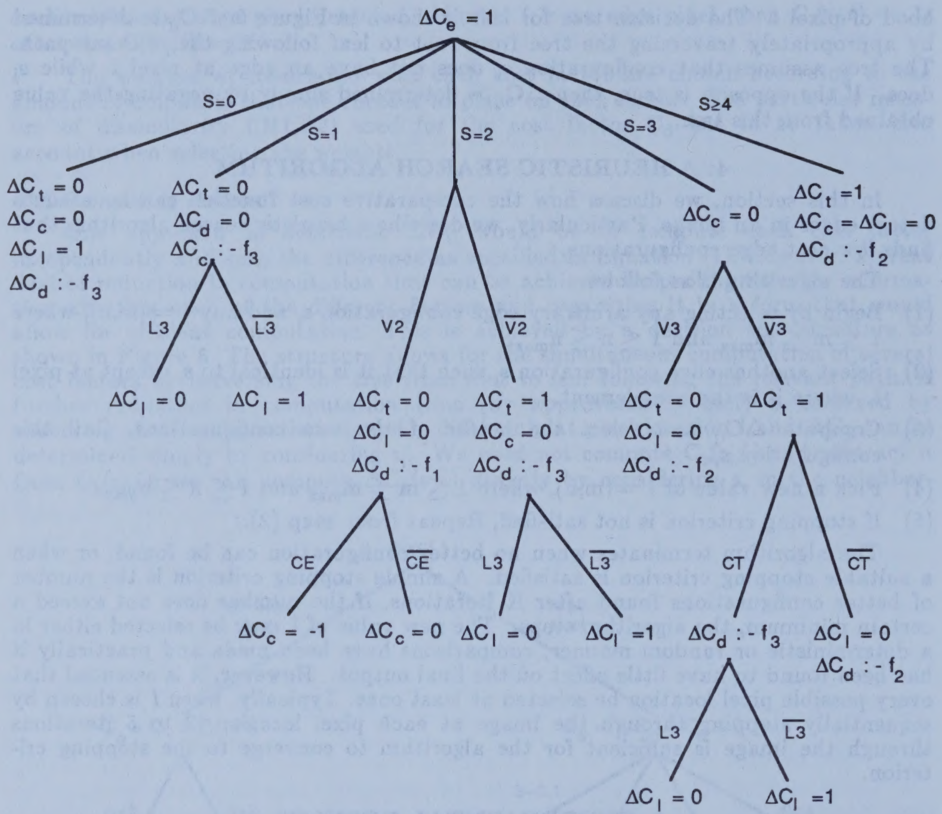
In this section, we present some results of the application of the heuristic algorithm to detect edges in images. Each image is 256×256 pixels in size, with gray level values between 0 and 255. Using the Pratt [8] figure of merit we compare the performance of this algorithm with the thresholded derivative of Gaussian operator and Sequential Edge Linking[3]. In each example, we started the algorithm using a random initial edge configuration, and proceeded to find better configurations using the technique described above. For our implementation, we used the value (2.0 1.1 1.1 1.0) for $(w_d\ w_t\ w_c\ w_e)$. With this, we disallow thick edges and reduce fragmentation by forcing edges to be connected when they are separated by one pixel. Also, the weight values of 2.0 and 1.0 for C_d and C_e respectively, will favor the detection of an edge point when $C_d \geq 0.5$. The cost factor C_l removes short edges, and the value of w_l was initially set to zero until the last few iterations when it was set equal to one.

Example 1: Ideal step edge

In this example we consider an artificial image which has a vertical step edge. Zero mean independently identically distributed Gaussian noise was added to the image. Figure 7(a) shows the noise corrupted step edge with a signal to noise ratio (SNR) of 0.25. Signal to noise ratio is defined as

$$\mathrm{SNR} = (h/\sigma_n)^2$$

where h is the edge step height and σ_n is the standard deviation of the Gaussian noise. We used a step height of 30; the gray levels on either sides of the edge were



$\Delta C_e = 1$
S=0
S=1
S=2
S=3
S≥4
$\Delta C_t = 0$
$\Delta C_c = 0$
$\Delta C_l = 1$
$\Delta C_d : - f_3$
$\Delta C_t = 0$
$\Delta C_d = 0$
$\Delta C_c : - f_3$
L3
$\overline{L3}$
$\Delta C_l = 0$
$\Delta C_l = 1$
V2
$\overline{V2}$
$\Delta C_t = 0$
$\Delta C_l = 0$
$\Delta C_d : - f_1$
$\Delta C_t = 1$
$\Delta C_c = 0$
$\Delta C_d : - f_3$
CE
$\overline{CE}$
$\Delta C_c = -1$
$\Delta C_c = 0$
L3
$\overline{L3}$
$\Delta C_l = 0$
$\Delta C_l = 1$
$\Delta C_c = 0$
V3
$\overline{V3}$
$\Delta C_t = 0$
$\Delta C_l = 0$
$\Delta C_d : - f_2$
$\Delta C_t = 1$
$\Delta C_t = 1$
$\Delta C_c = \Delta C_l = 0$
$\Delta C_d : - f_2$
CT
$\overline{CT}$
$\Delta C_d : - f_3$
$\Delta C_l = 0$
$\Delta C_d : - f_2$
L3
$\overline{L3}$
$\Delta C_l = 0$
$\Delta C_l = 1$

Figure 6. Decision tree for computing ΔC_{ij} assuming that pixel l in s_i is not an edge pixel.

110 and 140 respectively. We evaluate the performance of this operator using the Pratt figure of merit F, where

$$F = \frac{1}{I_M} \sum_{i=1}^{I_D} \frac{1}{1 + \alpha l_i^2} \cdot 100$$

where

$I_M = \max(I_D, I_I)$, I_I = number of ideal edge points, I_D = number of detected edge points, I_i = displacement of the ith detected edge point from the ideal edge, and α = scaling factor.

The value of F ranges from 0 to 100, with higher values indicating better performance. The value of 0.1 was used for α. Figures 7(b) - (d) show the output from three detection algorithms; thresholded derivative of Gaussian (∇G), sequential edge linking (SEL) and cost-minimization edge detection (CMED). A comparison using the Pratt figure of merit shows a value of F=73.2 for CMED, and a value of F=68.4 for SEL.

Example 2: House image

In this example we show results of the application of the algorithm to natural scenes. The original house image is shown in Figure 8(a). Zero mean additive Gaussian noise of standard deviation 35 was added to the image and the result is

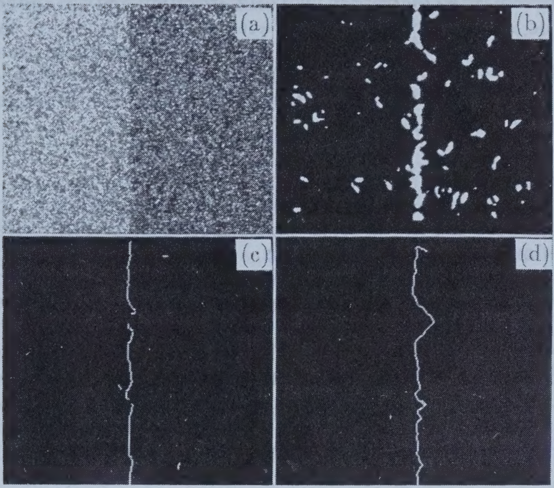


Figure 7.

(a) Ideal step edge with additive Gaussian noise. SNR=0.25.

(b) Edges from thresholded derivative of Gaussian operator.

(c) Edges from Cost Minimization Edge Detection algorithm (CMED).

(d) Edges from Sequential Edge Linking (SEL).

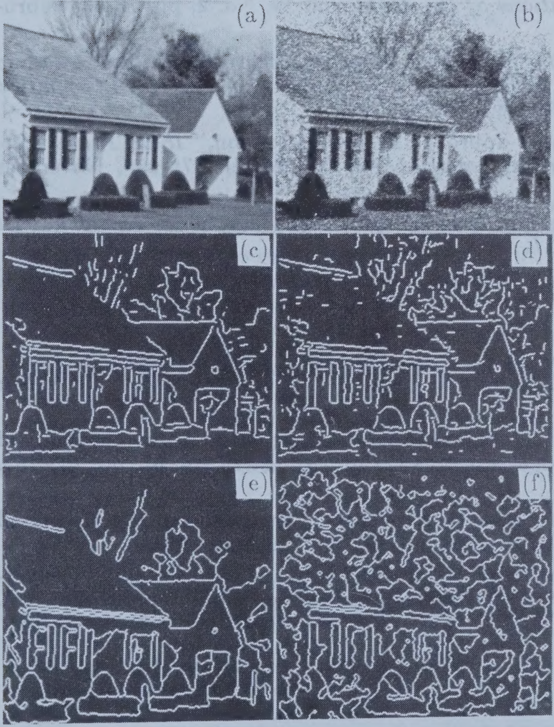


Figure 8.

(a) Original House image.

(b) House image with additive Gaussian noise of standard deviation = 35.

(c) Edges of original House image using CMED.

(d) Edges of noisy House image using CMED.

(e) Edges of original House image using SEL.

(d) Edges of noisy House image using SEL.

shown in Figure 8(b). We started the algorithm with a random initial edge configuration, and each new value of l (as discussed in Section 4) was selected by sequentially stepping through the pixels in a raster scan fashion. Figures 8(c) and 8(d) show the edges detected using CMED for the original and noisy images respectively. These results were obtained after 3 iterations through the image. Figures 8(e) and 8(f) show the edges detected using SEL.

6. CONCLUSIONS

In this paper, we have shown how edge detection can be cast as a problem in cost minimization. We have formulated a comparative cost function that evaluates the relative quality of a pair of similar edge configurations. Edges can be detected using a heuristic search algorithm; the detection process uses information from image data and local edge structure. Specifically, we have shown how edges can be detected by applying the algorithm to both artificial and real images. An extension of this approach to detect edges would be to formulate an absolute cost function that assigns an absolute cost value to any given edge configuration. The best edge configuration would be the one that achieves the global minimum of this cost function. Finding the global minimum would be an NP-complete problem. Simulated Annealing[9, 10] which is a stochastic optimization technique, could be used to find low cost edge configurations.

References

1. Michael Brady, "Computational approaches to image understanding," *Computing Surveys*, vol. 14, No. 1, pp. 3-71, March 1982.
2. Alberto Martellli, "An application of heuristic search methods to edge and contour detection," *Communications of the ACM*, vol. 19, No. 2, pp. 73-83, February 1976.
3. P. H. Eichel and E. J. Delp, "Sequential edge detection in correlated random fields," *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pp. 14-21, San Francisco, June 1985.
4. John Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. PAMI-8, No. 6, pp. 679-698, November 1986.
5. E. C. Hildreth, "Implementation of a theory of edge detection," *Massachusetts Inst. Technol., Cambridge, A.I. Memo 579*, 1980.
6. Robert M. Haralick, "Digital step edges from zero crossing of second directional derivatives," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. PAMI-6, No. 1, pp. 58-68, IEEE, January 1984.
7. Vishvjit Nalwa and Thomas O. Binford, "On edge detection," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. PAMI-8, No. 6, pp. 699-714, November 1986.
8. I. Abdou and W. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," *Proc. of IEEE*, vol. 67, pp. 753-763, May 1979.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, pp. 671-680, 13 May 1983.
10. Debasis Mitra, Fabio Romeo, and Alberto Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Adv. Appl. Prob.*, vol. 18, pp. 747-771, 1986.

REGULARIZED DIFFERENTIATION FOR EDGE DETECTION

AGGELOS K. KATSAGGELOS[1] and BARRY J. SULLIVAN[2]
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, Illinois 60208

SUMMARY

This paper describes the treatment of a problem in image processing using the theory of well-posed and ill-posed problems. A *well-posed* problem is one with a solution that is unique and depends continuously on the data. A problem that fails either of these two criteria is called *ill-posed.* The discretization of an ill-posed problem often leads to an *ill-conditioned* system of equations—i.e., a system in which small errors in the data can lead to large errors in the solution. *Regularization* refers to any mechanism for improving the condition of a system of equations before solving it.

It has been observed that many problems in image processing are ill-posed [1,2]. One in particular is the problem of edge detection. The ill-posed aspect of this problem is the differentiation step, which is an explicit operation in many schemes for finding edges in images. Differentiation is very sensitive to noise and its discrete version is an ill-conditioned operation. In this paper, we concentrate on regularizing the differentiation operation.

The regularization approach we take here is one first suggested by Miller [3]. In Miller regularization, the solution to a system of linear equations must satisfy two upper bounds: one on the residual, and one with respect to a constraint operator. The constraint operator is generally a p-th order derivative or a high-pass filter to guarantee some degree of smoothness in the solution. If sufficient information is available to determine these bounds, they can be combined into a single functional to be minimized. This approach has been previously applied to another ill-conditioned image processing problem, that of image restoration [4].

For the differentiation problem, we can relate a signal to its first derivative through an integral operator. If A is such an operator and g is the signal we wish to differentiate, solving

$$Af = g \tag{1}$$

for f is equivalent to finding the first derivative of g. The Miller formulation is then to find f such that

$$||g - Af|| \leq \epsilon \tag{2}$$

and

$$||Df|| \leq \delta \tag{3}$$

where ϵ represents the uncertainty or noise in g and D is an appropriate constraint operator. The value of δ is chosen based on the expected value of $||f||$. We discuss the effects of varying parameter values, as well as different forms of the matrix operators A and D.

We also consider two different approaches to finding f. The first follows Miller's suggestion to form the functional

$$M(\alpha, f) = ||g - Af||^2 + \alpha||Df||^2 \tag{4}$$

where $\alpha = (\epsilon/\delta)^2$, and minimize $M(\alpha, f)$ with respect to f. At low to moderate signal-to-noise ratios, this leads to a fairly well-conditioned system of equations. Subspace methods for solving systems of linear equations—such as the truncated SVD pseudoinverse [5]—can also be employed to extend the range of noise levels and further improve the solution.

A second approach centers about the convex sets

$$Q_1 = \{f \mid ||g - Af|| \leq \epsilon\} \tag{5}$$

and

$$Q_2 = \{f \mid ||Df|| \leq \delta\}. \tag{6}$$

Using the method of convex projections [6], we can arrive at a feasible solution $f \in Q_1 \cap Q_2$.

---

[1] The work of A. K. Katsaggelos was supported in part by the National Science Foundation, under Grant MIP-8614217.

[2] The work of B. J. Sullivan was supported in part by a grant from the Ameritech Foundation.

RESULTS

Several results have emerged from the experiments conducted to date. In these experiments, noise in the form of a random sequence (scaled to produce various signal-to-noise ratios) was added to samples from an analytic function to define the elements of g. As mentioned above, several forms for the A and D matrices and the parameter α have been tested. Experiments in which α was varied over a range of values suggest that setting α equal to $||n||/||g||$, which is essentially the inverse of the signal-to-noise ratio, yields a reasonable value.

For A, the integration matrix, an $n \times n$ lower triangular matrix was compared to an $(n-1) \times n$ matrix, where n is the length of the solution vector f. The non-zero entries of the $n \times n$ matrix were all set equal to the length of the sampling interval. The entries of the $(n-1) \times n$ matrix were based on trapezoidal and Simpson's rule integration formulas. (The rectangular form of the second matrix is due to the need for at least two points for trapezoidal rule integration.) The underdetermined nature of Equation (1) with the $(n-1) \times n$ matrix leads to difficulties in minimizing $M(\alpha, f)$ when little or no noise is present. At lower signal-to-noise ratios (and consequently higher values of α), these problems disappear. Below a signal-to-noise ratio of 30 dB, the $(n-1) \times n$ formulation for A yields slightly better results.

The different forms for D were all based on central difference rules for differentiation. First, second, third and fourth derivatives were all considered as possible constraint operators. It was found that a second-derivative form for D gave the best results.

REFERENCES

[1] T. Poggio, "Early vision: From computational structure to algorithms and parallel hardware," *Comp. Vis. Graph. Image Proc.*, vol. 31, pp.139–155, 1985.

[2] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 26 Sept. 1985.

[3] K. Miller, "Least-squares method for ill-posed problems with a prescribed bound," *SIAM J. Math. Anal.*, vol. 1, pp. 52–74, Feb. 1970.

[4] A. K. Katsaggelos, J. Biemond, R. M. Mersereau and R. W. Schafer, "A general formulation of constrained iterative restoration algorithms," *Proc. 1985 Int. Conf. Acoust., Speech, Signal Proc.*, pp. 700–703, Tampa, FL, March 1985.

[5] B. J. Sullivan and B. Liu, "On the use of singular value decomposition and decimation in discrete-time band-limited signal extrapolation," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-32, no. 6, pp. 1201–1212, Dec. 1984.

[6] D. C. Youla and H. Webb, "Image restoration by the method of convex projections," *IEEE Trans. Med. Imag.*, vol. MI-1, pp. 81–94, Oct. 1982.

EDGE ENHANCEMENT USING ORDER STATISTIC LAPLACIAN[*]

YONG HOON LEE and SOON YOUNG PARK
Department of Electrical and Computer Engineering
State University of New York at Buffalo
Buffalo, New York 14260
Tel. (716) 636-2406

ABSTRACT

One- and two-dimensional continuous and discrete convex/concave (C/C) edges, which are useful blurred edge models, are defined and characterized. In particular, a necessary and sufficient condition for an edge to be C/C is given. It is observed that the well-known edge enhancing operator based on the Laplacian implicitly assumes that the blurred edge is C/C. Some edge enhancing operators based on a modification of the Laplacian called the order statistic (OS) Laplacian are introduced. It is shown that the edge enhancing operators employing the Laplacian or the OS Laplacian enhance 2D C/C edges. The statistical properties of these operators are analyzed for input signals with Gaussian density functions.

I. INTRODUCTION

In many cases, the restoration of blurred images has been carried out through linear deconvolution. While linear deconvolution procedures are very effective in deblurring, they are rather hard to implement because sufficient knowledge of a model representing the blur is required. Image sharpening or edge enhancing techniques are simple image enhancement tools counteracting blur without the knowledge about the blur. One of the edge enhancing techniques, and probably the most widely used one, is $F - \nabla^2 F$ where F is an image and $\nabla^2 F$ is the Laplacian operator[**] [1]. Because the edge enhancing techniques that we shall consider are based on $\nabla^2 F$, the reason why $F - \nabla^2 F$ has edge enhancing effect is worth describing here.

While a number of rationales for the relevance of $F - \nabla^2 F$ to edge enhancement have been given in Rosenfeld and Kak [1], we can have further insight into the edge enhancing operator by considering a typical one-dimensional (1D) blurred edge shown in Fig. 1. In Fig. 1(a), the increasing edge curve is bending upward at the lower side of the edge, and bending downward at the higher side of the edge. It is important to note that, in general, the second derivative of a curve bending upward is positive, while that of a curve bending downward is negative. Thus, $F - \nabla^2 F$ is generally less than F at the lower side of the edge, and larger than F at the higher side. This results in edge enhancement. The second derivative of the edge in Fig. 1(a), and the enhanced edge are shown in Fig. 1(b) and (c), respectively. In general, a curve bending upward is convex, and a curve bending downward is concave. In many practical situations, an increasing edge is

---

[*] This work was supported by the National Science Foundation under Grant DCI-8611859.

[**] The image F is 1D or 2D, continuous or discrete. When it is necessary to distinguish a continuous image from a discrete one, continuous and discrete images are denoted by F_a and F, respectively. When F is 1D and continuous, $\nabla^2 F = d^2F/dx^2$.

composed of an increasing convex part followed by a concave one, while a decreasing edge is composed of a decreasing concave part followed by a convex one. Such edges are called the *convex/concave* (C/C) edges. [2]. In essence, $F - \nabla^2 F$ implicitly assumes that the edge to be enhanced is C/C.

The major difficulty in applying the Laplacian to images is its sensitivity to noise. To reduce the noise sensitivity, the Laplacian is often used after some pre-filtering. For example, edge-preserving filters such as the median filter [1], [3], may be used before applying the Laplacian. Alternatively, some modifications of the Laplacian may be used [1]. One such operator which we call the *order statistic* (OS) Laplacian is an operator propotional to the difference between the average and the median, computed over the same neighborhood of the given point. The modified Laplacian operators are less sensitive to noise than the Laplacian.

Recently, Lee and Fam [2] introduced 1D discrete C/C edge model, and an edge enhancing operator called the *comparison and selection* (CS) filter was derived based on this edge model. It can be seen that the CS filter selects one of the sample values inside the window depending on the sign of the OS Laplacian.

In this paper, we first extend some of the results in [2]. Specifically, 1D and 2D, continuous and discrete C/C edge models are defined and characterized in Section II. Then in Section III, edge enhancing properties of some operators employing the Laplacian and the OS Laplacian are studied using the 2D discrete C/C edge model. The noise sensitivities of the edge enhancing operators are investigated statistically in Section IV. Finally, in Section V some experimental results are presented. Throughout this paper, our results are presented for nondecreasing edges but extending them to nonincreasing edges is straightforward and will not be discussed.

II. CONVEX/CONCAVE EDGES

A. *One-Dimensional Convex/Concave Edges*

Definition 1: A real-valued function $F(x)$ defined in $(-\infty, \infty)$ is a nondecreasing edge if (i) $F(x)$ is a nondecreasing function with $\lim_{x \to -\infty} F(x) = 0$ and $\lim_{x \to \infty} F(x) = c$ for some positive constant c, (ii) $F(x)$ is either continuous or piecewise continuous, and (iii) $F(x)$ has derivative, $F'(x)$, except at some isolated points, i.e., $F(x)$ is piecewise differentiable.

An ideal step edge is modeled by the unit step function $U(x)$ which is zero if $x < 0$, and is one otherwise. The C/C edge is defined as follows:

Definition 2: A continuous nondecreasing edge $F(x)$ is C/C if $F(x)$ is convex if $x \leq x_0$, and concave otherwise, for some x_0.

The definitions of convex and concave functions may be found in [4]. The point x_0 is called the *inflection* point of the C/C edge. The inflection point may not be determined uniquely for some C/C edges such as ramp edges. Whenever this happens, we select the rightmost point among the possible points as the inflection point x_0.

A C/C edge can be obtained by convolving an ideal step edge with a proper function which we call the C/C edge *generating* function. In other words, if an ideal step edge is passed through the linear time invariant system whose impulse response is a C/C edge generating function then a C/C edge is produced. Next this observation is described following the definition of the C/C edge generating function.

Definition 3: A nonnegative real-valued function $W(x)$ which is neither a constant nor a delta function is a C/C edge generating function if (i) $\lim_{x \to -\infty} W(x) = \lim_{x \to \infty} W(x) = 0$, (ii) $W(x)$ is either continuous or piecewise continuous, (iii) $W(x)$ is either differentiable or piecewise differentiable, and (iv) $W(x)$ is nondecreasing if $x \leq x_0$, and nonincreasing, otherwise, for some constant x_0.

Property 1: A necessary and sufficient condition for a function $F(x)$ to be a C/C edge is that there exists a C/C edge generating function $W(x)$ for which $F(x) = U(x) * W(x)$ where "*" represents the convolution.

The proof of this property is rather long and thus omitted. This property shows a relationship between the edge enhancement and the deconvolution: Any edge enhancing operator that can produce an ideal step edge from a C/C edge can be thought of as a deconvolving operator. Examples of the C/C edges obtained through convolutions are shown in Fig. 2.

In [2], the definition of the 1D discrete C/C edge follows the definitions of the discrete convexity and concavity. In the following, unlike [2], the 1D discrete C/C edge is defined as a sequence derived from a 1D continuous C/C edge by periodic sampling. We do this merely because the extension to 2D discrete C/C edges can be done easily in this manner. We shall see that the definition of the 1D discrete C/C edge in [2] is closely related to the one presented here.

Definition 4: The sequence obtained from a 1D continuous edge by periodic sampling is said to be a 1D discrete edge. More precisely, if $F_a(x)$ is a continuous edge, then the discrete edge $F(m)$ is given by

$$F(m) = F_a(mT) \tag{1}$$

where m is an integer, $-\infty < m < \infty$, and T is the sampling period. The discrete edge is C/C whenever $F_a(x)$ is C/C.

It is noted that periodic sampling of a C/C edge may result in a step sequence modeling discrete ideal step edges. Thus discrete ideal step edges are considered as a discrete C/C edge.

Property 2: Suppose that $F(m)$ is a discrete C/C edge. Then $F(m)$ has the following two properties:

(a) There exists a point m_0 for which $D(m-1) \leq D(m)$ if $m \leq m_0$, and $D(m) \geq D(m+1)$ otherwise, where $D(m) = F(m) - F(m-1)$.

(b) $F(m) = U(m) * D(m)$ where "*" denotes the discrete convolution, and $U(m)$ is zero if $m < 0$, and is one otherwise.

Proof: (a) Let i be the sampling point right before the inflection point x_0 of $F_a(x)$. (If a value is sampled at x_0, then $i = x_0$.) From the convexity and the concavity of $F_a(x)$ it follows that

$$F_a(i-kT) - F_a(i-kT-1) \geq F_a(i-kT-1) - F_a(i-kT-2) \tag{2a}$$

$$F_a(i+kT+2) - F_a(i+kT+1) \geq F_a(i+kT+3) - F_a(i+kT+2) \tag{2b}$$

for any nonnegative integer k [4]. One of the points i, $i + T$, $i + 2T$ becomes m_0 depending on the derivatives of $F_a(x)$. The proof of (b) is trivial.

In [2], a sequence satisfying the conditions in the part (a) of this property is defined as a discrete C/C edge. The point m_0 is called the inflection point of the discrete C/C edge.

B. Two-Dimensional Convex/Concave Edges

The 2D edge is defined as a surface which is constant in one direction, and its profile taken in the edge orientation which is the direction perpendicular to the direction of invariance is a 1D edge, as shown in Fig. 3. The edge orientation is represented by θ that is the angle measured from the positive half of the x-axis. For simplicity, our results are presented for $0 \leq \theta < \pi/2$, but they can be extended to θ outside the range in a straightforward manner, and will not be discussed.

Let the edge with orientation θ be described by the function $F_\theta(x,y)$. We define a shifted and rotated coordinated system, the $(x_{\phi bc}, y_{\phi bc})$ system, as

$$x_{\phi bc} = (x - b)\cos\phi + (y - c)\sin\phi \tag{3a}$$

$$y_{\phi bc} = -(x - b)\sin\phi + (y - c)\cos\phi \tag{3b}$$

where b and c represent the amounts of shifting in x- and y-directions, respectively, and ϕ is the angle of rotation, $0 \leq \phi \leq 2\pi$. It is seen that the orientation of the edge $F_\theta(x,y)$ is parallel to the $x_{\theta bc}$-axis and normal to $y_{\theta bc}$-axis. The profile of $F_\theta(x,y)$ taken in the orientation ϕ at the point (b,c) can be expressed as

$$P_F(x_\phi) = F_\theta(x_\phi\cos\phi+b, x_\phi\sin\phi+c) \tag{4}$$

where we write x_ϕ instead of $x_{\phi bc}$ to simplify the notation. The 2D edge with orientation θ is defined formally as follows:

Definition 5: A real-valued function $F_\theta(x,y)$, $-\infty < x, y < \infty$, is a 2D edge with orientation θ if the profile $P_F(x_\phi)$ with $\phi = \theta + \pi/2$ is constant, and the profile $P_F(x_\theta)$ is a 1D edge for any constants b and c. In addition, if $P_F(x_\theta)$ is a 1D C/C edge, then $F_\theta(x,y)$ is a 2D C/C edge with orientation θ.

A 2D ideal step edge with orientation θ is modeled by the 2D edge $U_\theta(x,y)$ given by

$$U_\theta(x,y) = \begin{cases} 0, & \text{if } y \leq -x/\tan\theta \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

As with the 1D case, the 2D C/C edge can be generated through the 2D convolution of $U_\theta(x,y)$ with a proper function.

Property 3: A necessary and sufficient condition for a real-valued function $F(x,y)$ is a 2D C/C edge with orientation θ is that there exists a function $W_\theta(x,y)$ for which $F(x,y) = U_\theta(x,y) ** W_\theta(x,y)$ where $W_\theta(x,y)$ is zero if $y \neq x\tan\theta$ and its profile taken in the orientation θ at a point (b,c), which is represented by $P_W(x_\theta) = W_\theta(x_\theta\cos\theta+b, x_\theta\sin\theta+c)$ is a 1D C/C edge generating function, and ** represents the 2D convolution.

Proof: Suppose that there exists $W_\theta(x,y)$ for which $F(x,y) = U_\theta(x,y) ** W_\theta(x,y)$. Then $F(x,y) = \int_{-\infty}^{g(x,y)} P_W(\alpha)d\alpha$, where $g(x,y) = (x + sy)/(s^2 + 1)$, and $s = \tan\theta$. It is seen that $P_F(x_\phi)$ with $\phi = \theta + \pi/2$ is constant, and $P_F(x_\theta) = \int_{-\infty}^{x+\beta} P_W(\alpha)d\alpha$ with $\beta = (c - bs)s/(s^2 + 1)$. From the proof of Property 1, $P_F(x_\theta)$ is a 1D C/C edge. Thus $F(x,y)$ is a 2D C/C edge with orientation θ. The converse can be proved as in Property 1.

We now observe that a profile of a 2D C/C edge taken in an arbitrary direction is a 1D C/C edge if it is not a constant.

Property 4: A 2D edge, $F_\theta(x,y)$, with orientation θ is C/C if and only if the profile $P_F(x_\phi)$ is 1D C/C for any ϕ, $0 \le \phi \le 2\pi$, and $\phi \ne \theta \pm \pi/2$.

Proof: Suppose that $F_\theta(x,y)$ is C/C. Then $P_F(x_\theta)$ is a 1D C/C edge. Consider $P_F(x_\phi)$ with a ϕ, $0 \le \phi \le 2\pi$. Assume that $P_F(x_\theta)$ and $P_F(x_\phi)$ are taken at the origin ($b = c = 0$ in (3)). Since $F_\theta(x,y)$ is constant in the direction $\theta + \pi/2$, we get $P_F(x_\phi) = P_F(x_\theta)$ given $x_\theta = tx_\phi$ where $t = \sin\theta\sin\phi + \cos\theta\cos\phi$. Hence $P_F(x_\phi)$ is convex if $x_\phi \le x_{\theta 0}/t$, and concave otherwise, where $x_{\theta 0}$ is the inflection point of the 1D C/C edge $P_F(x_\theta)$. Similarly, we can show the case where the profiles are taken at an arbitrary point. The proof of the converse is trivial.

In Section III, we shall see that this property and the discrete version of this (Property 5(b)) are useful in examing the properties of edge enhancing operators.

Unlike the continuous case, the 2D discrete edge cannot be defined for all orientations. This is because the orientation in discrete domain is given by the angle

$$\theta = \tan^{-1}(v_\theta/h_\theta) \tag{6}$$

where v_θ and h_θ must be integers. (θ is the angle for which $\tan\theta$ is a rational number.) We always assume that the orientation of the 2D discrete edge is given by (6). Let the 2D discrete edge with orientation θ be described by the array $F_\theta(m,n)$, $-\infty < m, n < \infty$. A shifted and rotated discrete coordinated system, the (m_ϕ, n_ϕ) system, is defined as

$$m_{\phi ij} = (m - i)v_\phi + (m - j)h_\phi \tag{7a}$$

$$n_{\phi ij} = -(m - i)h_\phi + (n - j)v_\phi \tag{7b}$$

where v_ϕ and h_ϕ are the smallest positive integers satisfying $\tan\phi = v_\phi/h_\phi$ and i, j are integers representing the amount of shifting. The profile of $F_\theta(m,n)$ taken in the orientation ϕ at the point (i,j) can be expressed as

$$P_F(m_\phi) = F_\theta(m_\phi h_\phi + i,\ m_\phi v_\phi + j) \tag{8}$$

where ij is dropped from $m_{\phi ij}$ to simplify the notation.

As with the 1D case, the 2D discrete edge is derived from the 2D continuous edge by periodic sampling.

Definition 6: If $F_{a\theta}(x,y)$ represents a continuous edge with orientation θ where θ is given by (6), then the discrete edge $F_\theta(m,n)$ with orientation θ is given by

$$F_\theta(m,n) = F_{a\theta}(mT, nT) \tag{9}$$

where m and n are integers, $-\infty < m, n < \infty$. The discrete edge is C/C whenever the continuous edge is C/C.

The 2D discrete ideal step edge which is considered as a special case of 2D discrete C/C edges is modeled by

$$U_\theta(m,n) = \begin{cases} 0, & \text{if } n < -mh_\theta/v_\theta \\ 1, & \text{otherwise.} \end{cases} \qquad (10)$$

Property 5: Let $F_\theta(m,n)$ be a 2D discrete C/C edge with orientation θ. Then $F_\theta(m,n)$ has the following properties:

(a) $F_\theta(m,n) = U_\theta(m,n) ** W_\theta(m,n)$ where "**" denotes the 2D discrete convolution, and $W_\theta(m,n)$ is zero if $n \neq mv_\theta/h_\theta$ and its profile taken in the orientation θ at a point (i,j), which is represented as $P_W(m_\theta) = W_\theta(m_\theta h_\theta + i, m_\theta v_\theta + j)$, is given by $P_W(m_\theta) = P_F(m_\theta) - P_F(m_\theta - 1)$.

(b) The profile $P_F(m_\phi)$ taken in any orientation ϕ, $0 \leq \phi \leq 2\pi$, $\phi \neq \theta \pm \pi/2$, is a 1D discrete C/C edge.

Part (a) can be proved as in the proof of Property 3, and Part (b) is a direct consequence of Property 4.

III. ENHANCEMENT OF CONVEX/CONCAVE EDGES

In this section, we show that the operators employing the Laplacian and the OS Laplacian can enhance 2D discrete C/C edges. First the OS Laplacian which is a modification of the digital Laplacian is defined.

The digital Laplacian of an image at point (i,j) is commonly implemented by

$$\nabla^2 F(i,j) = L[A(i,j) - F(i.j)] \qquad (11)$$

[1] where L is the number of samples inside the window centered at (i,j) and $A(i,j)$ is the average of the samples inside the window. The OS Laplacian, denoted by $OS\nabla^2 F$, is defined by

$$OS\nabla^2 F(i,j) = L[A(i,j) - M(i,j)] \qquad (12)$$

where $M(i,j)$ is the sample median among the values inside the window. Clearly, the OS Laplacian is equal to the digital Laplacian whenever the center value of the window, $F(i,j)$, equals the median, $M(i,j)$. We can see that the OS Laplacian is a linear combination of the ordered values inside the window. So the OS Laplacian is a special case of the OS filter introduced by Bovik et. al. [5].

The window centered at the point (i,j), say W_{ij}, is defined in terms of a neighborhood pixel locations. The size of a window is the total number of pixel locations in it. We always assume that the window W_{ij} includes the center point (i,j) and is symmetric with respect to the center. For such window, the odd window size is guaranteed, and thus the median value can always be selected.

The edge enhancing operators that we shall consider include $F - \nabla^2 F$, $M - \nabla^2 M$ which is median filtering followed by $F - \nabla^2 F$, $F - OS\nabla^2 F$ and CS filters. These edge enhancing operators enhance C/C edges if the Laplacian or the OS Laplacian is positive in convex regions, and is negative in concave regions. It should be noted that the Laplacian and the OS Laplacians of a C/C edge are the same. This can be seen easily for 1D C/C edges. The 2D case can be shown as follows: Let $F(m,n)$ be a 2D C/C edge and $L_{ij}(\phi)$ be the line passing through the points (i,j) and $(i+h_\phi, j+v_\phi)$ where $\tan\phi = v_\phi/h_\phi$. Since the profile taken in the orientation ϕ is either 1D C/C or constant from Property 5(b), we get

$$\text{Median}\ \{F(m,n) \mid (m,n) \in L_{ij}(\phi) \cap W_{ij}\} = F(i,j)$$

for each ϕ. Thus the median $M(i,j)$ selected from W_{ij} is equal to $F(i,j)$ which is the center value of the window [3]. This indicates that the 2D C/C edge is invariant to 2D median filtering with a symmetric window W_{ij}, and $\nabla^2 F = \nabla^2 M = \text{OS}\nabla^2 F$. In the rest of this section, we will study the C/C edge enhancing properties of these operators by examining the sign of the (OS) Laplacian.

For 1D C/C edges some results regarding the sign of the OS Laplacian are obtained [2]. The results are extended to 2D case as follows:

Property 6: Suppose that a symmetric window W_{ij} moves over a 2D C/C edge. Then $A(i,j) \geq M(i,j)$ if all samples inside W_{ij} are from the convex region, and $A(i,j) \leq M(i,j)$ if all of the samples are from the concave region.

Proof: Since $M(i,j) = M_\phi(i,j) = F(i,j)$, we get

$$A(i,j) - M(i,j) = \sum_\phi n_\phi [A_\phi(i,j) - M_\phi(i,j)]/(2N+1)^2$$

where

$$M_\phi(i,j) = \text{Median}\ \{F(m,n) \mid (m,n) \in L_{ij}(\phi) \cap W_{ij}\}, \qquad (13a)$$

$$A_\phi(i,j) = \text{Average}\{F(m,n) \mid (m,n) \in L_{ij}(\phi) \cap W_{ij}\}, \qquad (13b)$$

n_ϕ is the number of samples in $L_{ij}(\phi) \cap W_{ij}$, and ϕ ranges from 0 to 2π. From property 5(b) and [2], $A_\phi(i,j) \geq M_\phi(i,j)$ in the convex region, and the inequality is reversed in the concave region for each ϕ. This completes the proof.

This property implies that the edge enhancing operators employing the OS Laplacian enhance 2D C/C edges. For 2D C/C edges with certain orientations and some 2D windows, we can show that in each scan line there exists only one point at which the sign of the OS Laplacian is changed.

Property 7: Let $F_\theta(m,n)$ be a 2D discrete C/C edge with orientation θ and a symmetric window W_{ij} scan $F_\theta(m,n)$ in the horizontal direction. If either W_{ij} is square and $\theta = 0$, or W_{ij} is cross and $\theta = 0$ or $\pi/4$, then there exists an integer k for which $A(i,j) \geq M(i,j)$ if $i \leq k$, and $A(i,j) \leq M(i,j)$ if $i > k$ for each j.

Proof: If the $(2N+1)\text{x}(2N+1)$ square window is employed and $\theta = 0$, then $A(i,j) - M(i,j) = [A_0(i,j) - M_0(i,j)]$ where $A_0(i,j)$ and $M_0(i,j)$ are given by (13) with $\phi = 0$. If the $(2N+1)\text{x}(2N+1)$ cross window is employed then $A(i,j) - M(i,j) = C[A_0(i,j) - M_0(i,j)]$ with $C = (2N+1)/(4N+1)$ if $\theta = 0$, and $C = 2(2N+1)/(4N+1)$ if $\theta = \pi/4$. Hence for each case the 2D problem is reduced to a 1D problem and the property holds by Property 5(b) and the result in [2].

The property similar to this holds when the window scans the edge in the vertical direction. This property prevents the edge enhancing operators from causing some oscillations around the inflection point m_0.

VI. NOISE SENSITIVITY

In this section, we study the noise sensitivity properties of $F - \nabla^2 F$, $M - \nabla^2 M$, $F - \text{OS}\nabla^2 F$ and CS filters by comparing their output variances when the input signal $F(i,j)$ is independently and identically distributed (i.i.d.) with the Gaussian density function. The procedures for obtaining the output variances of $M - \nabla^2 M$ and $F - OS\nabla^2 F$ follow. (It is trivial to obtain the output variance of $F - \nabla^2 F$, and that of the CS filter has been obtained in [6].) If we denote the results of $M - \nabla^2 M$ and $F - OS\nabla^2 F$ by $G_M(i,j)$ and $G_0(i,j)$, respectively, then they are given by

$$G_M(i,j) = M(i,j) + LM(i,j) - \sum_{(i-k,\,j-m)\in W_{ij}} M(i-k,j-m) \qquad (14)$$

$$G_0(i,j) = F(i,j) + LM(i,j) - \sum_{(i-k,\,j-m)\in W_{ij}} F(i-k,j-m) \qquad (15)$$

It is not difficult to see that $M - \nabla^2 M$ and $F - OS\nabla^2 F$ are unbiased estimators of the mean if the input density is symmetric. (The unbiasedness of the CS filter is shown in [6].) Therefore when the input has zero mean the variances of $M - \nabla^2 M$ and $F - OS\nabla^2 F$ are given by $\sigma_M^2 = E[G_M^2(i,j)]$ and $\sigma_0^2 = E[G_0^2(i,j)]$. The evaluation of σ_M^2 and σ_0^2 requires the joint probability density function (pdf) of median values which can be found in [7]. Using the joint pdf in [7], we numerically evaluate σ_M^2 and σ_0^2 for the Gaussian input with zero mean and variance one. The results associated with the 3x3 square and cross windows are tabulated in Table 1. For comparison, the output variances of $F - \nabla^2 F$ and CS filter are also shown. The results show that only the CS filter can reduce noise components, while the others amplify them. As expected, $F - \nabla^2 F$ severely amplifies noise, and $M - \nabla^2 M$ and $F - OS\nabla^2 F$ are much less sensitive to noise than $F - \nabla^2 F$. Between $M - \nabla^2 M$ and $F - OS\nabla^2 F$, the latter is slightly less sensitive to noise than the former. Although $M - \nabla^2 M$ and $F - OS\nabla^2 F$ amplify noise, sometimes they can produce more subjectively pleasing image than the CS filter. This is because $M - \nabla^2 M$ and $F - OS\nabla^2 F$ accentuate edges, as will be seen in section V.

V. SIMULATION RESULTS

In this section, edge enhancing operators are applied to 1D signals with blurred edges to illustrate their performance characteristics. Fig. 4 (a) shows a signal with blurred edges corrupted by additive white Gaussian noise with mean zero and variance two. Fig. 4 (b)-(e) exhibit the results of $F - \nabla^2 F$, $F - OS\nabla^2 F$, $M - \nabla^2 M$ and CS filtering with $J = 1$ of the noisy blurred signal when the window size is five. Clearly, the CS filter tends to produce ideal step edges while suppressing noise. $M - \nabla^2 M$ and $F - OS\nabla^2 F$ resulted in undershooting and overshooting, respectively, at the lower sides and at the higher sides of the edges like $F - \nabla^2 F$, but they amplified noise less than $F - \nabla^2 F$. The undershooting and overshooting result in accentuated edges, and sometimes images with accentuated edges are more subjectively pleasing [8]. Therefore $M - \nabla^2 M$ and $F - OS\nabla^2 F$ are useful alternatives to $F - \nabla^2 F$ in image enhancement. On the other hand, the experimental results in [2] indicate that the CS filter is useful as a prefilter before edge detection.

REFERENCES

1. A. Rosenfeld and A. C. Kak, Digital Picture Processing, 2nd ed. New York: Academic, 1982.
2. Y. H. Lee and A. T. Fam, "An edge gradient enhancing

adaptive order statatistic filter," IEEE Trans. Acoust., Speech Signal processing, vol. ASSP-35, pp. 680-695, MAY 1987.

3. T. S. Huang, Ed., Two-Dimensional Digital Signal Processing II: Transforms and Median Filters. New York: Springer-Verlag, 1981.
4. W. Rudin, Principles of Mathematical Analysis, 3rd ed. Mcgraw-Hill, 1976.
5. A. C. Bovik, T. S. Huang, and D. C. Munson, Jr., "A generalization of median filtering using linear combinations of order statistics," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, pp. 1342-1350, Dec. 1983.
6. Y. H. Lee and S. Y. Park, "Some statistical properties of comparison and selection filters," In preparation.
7. F. Kuhlman and G. L. Wise, "On second moment properties of median filtered sequences of independent data," IEEE Trans. Commun., vol. COM-29, pp.1374-1379, Sep. 1981.
8. W. K. Pratt, Digital Image Processing, New York: Wiley, 1978.

Operator		Output variance	
		Square	Cross
$F - \nabla^2 F$		89.0	29.0
$F - OS\nabla^2 F$		5.456	3.170
$M - \nabla^2 M$		6.648	5.570
CS	J = 1	0.281	0.548
	J = 2	0.505	1.414
	J = 3	0.955	
	J = 4	2.135	

Table 1. Output variances of the operators when window size is 3x3 and the input variance is one.

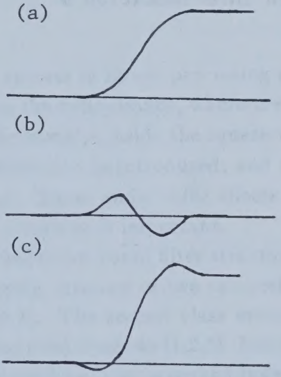


Fig. 1. (a) A convex/concave edge, F(x). (b) d^2F/dx^2. (c) $F - d^2F/dx^2$.

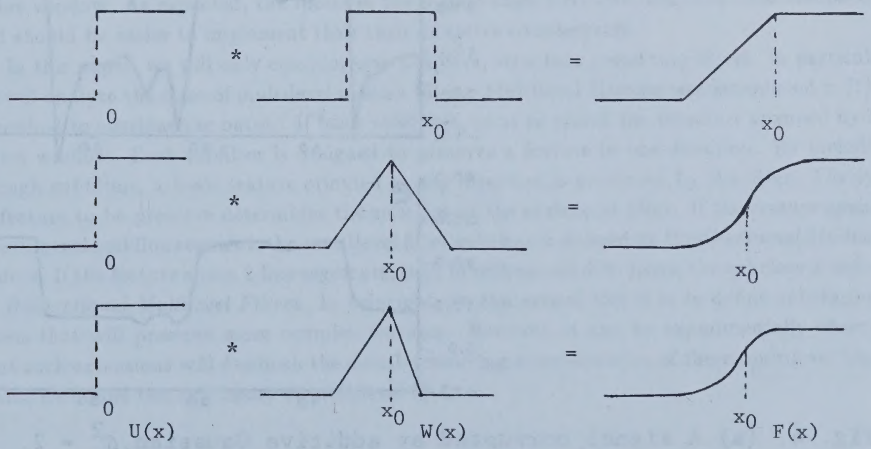


Fig. 2. Convex/concave edges.

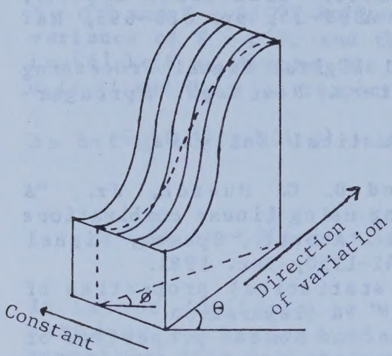

Direction of variation
Constant
θ
ø

Fig. 3. A 2D convex/concave edge with orientation θ. The dotted line shows the profile taken in orientation ø.

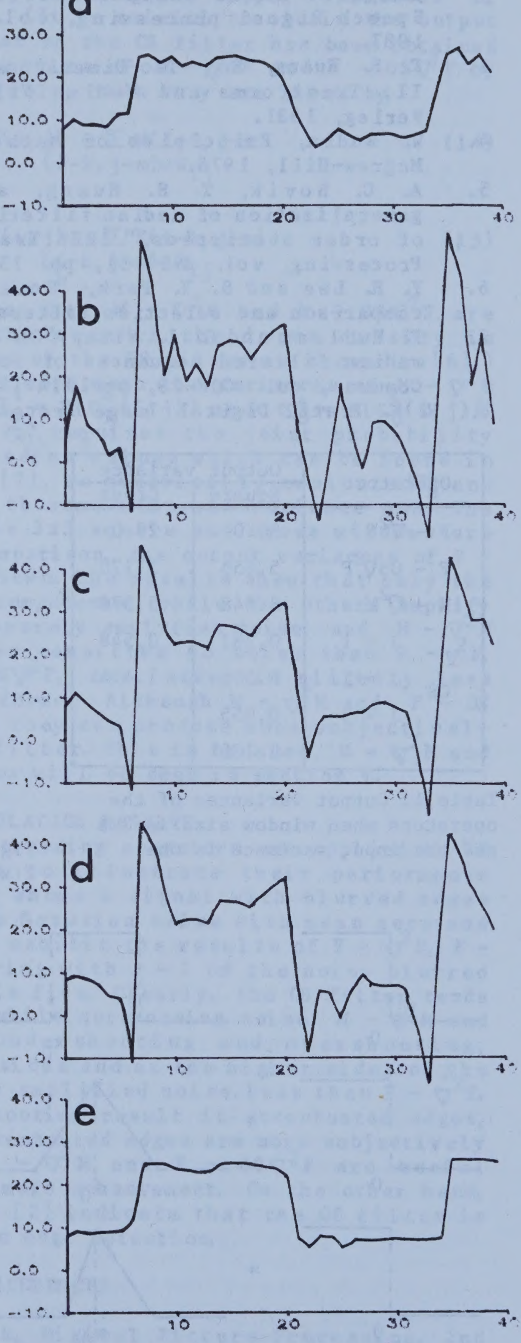


Fig. 4. (a) A signal corrupted by additive Gaussian, $\sigma^2 = 2$. Results of (b) $F - \nabla^2 F$, (c) $F - OS\nabla^2 F$, (d) $M - \nabla^2 M$, (e) CS, J=1, when 2N + 1 = 5.

MULTILEVEL MEDIAN FILTERS

Gonzalo R. Arce and Russell E. Foster
Department of Electrical Engineering
University of Delaware
Newark, Delaware 19716

Abstract

In this paper, the theoretical analysis of multilevel median filters is developed. It is shown that multilevel Multilevel median are very effective in preserving image-detail as well as in attenuating noise. It is shown that multilevel median filters are a combination of max/median and min/median filters. Statistical threshold decomposition is applied to derive the statistical characteristics of these filters, and the results are used to evaluate the performance of this class multilevel filters. Finally, a quantitative and qualitative comparison of the multilevel filters and of other efficient detail-preserving filters is presented.

1 Introduction

Two-dimensional median filters have been used with some success in image processing applications. A $2N+1 \times 2N+1$ running window is scanned across the entire image, where the output associated with the center of the window is the median of the samples inside the square window. Although noise suppression is obtained, too much signal distortion is introduced, and features such as thin lines and other fine details are distorted or lost. These undesirable effects are not acceptable in many applications where the preservation of structure is important.

Recently, several new detail-preserving, median and ranked order-based filter structures have been proposed in the literature [1-8]. These filters fall, roughly, into one of two categories. The first class is the set of adaptive ranked order filters [3,4,6,8]. The second class encompasses structure-preserving ranked-order filters with fixed (non-adaptive) windows [1,2,5]. Local statistics are used extensively to specify the samples to be considered by adaptive ranked-order filters. The principal advantage of non-adaptive filters is that neither, a-priori information of the image is required such as minimum edge height, nor do we need to compute local statistics inside the filters window. As expected, the filters in the second class have less computational complexity, and should be easier to implement than their adaptive counterparts.

In this paper, we will only consider non-adaptive, structure preserving filters. In particular, we will analyze the class of multilevel median filters. Multilevel filtering was introduced in [1] as a method to combine the output of basic subfilters, so as to match the structure spanned by the filters window. Each subfilter is designed to preserve a feature in one direction. By including enough subfilters, a basic feature oriented in any direction is preserved by the filter. The type of feature to be preserve determines the subclass of the multilevel filter. If the feature spans a one dimensional line segment, the multilevel filter subclass is defined as *Unidirectional Multilevel Filters.* If the feature spans 2 line segments, each in orthogonal directions, the subclass is defined as *Bidirectional Multilevel Filters.* In principle, we can extend this idea to define subclasses of filters that will preserve more complex features. However, it can be experimentally observed that such extensions will diminish the detail-preserving characteristics of these multilevel filters. Thus, we define two subclasses of multilevel filters.

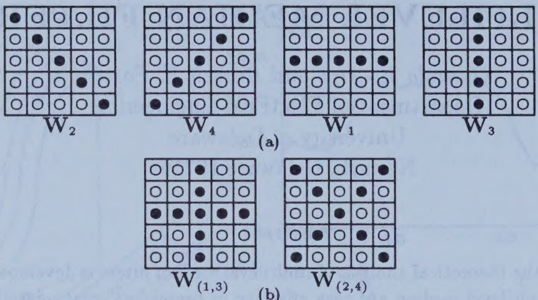


Figure 1: Subwindows used by multilevel filters: (a) Unidirectional; (b) Bidirectional .

A. Unidirectional Multilevel Filters

Let $\{a(.,.)\}$ be a discrete 2-dimensional sequence $\{a(n_1,n_2);\ n_1,n_2 \in \mathbf{Z}\}$ where the index $\mathbf{Z}$ is the set of all integers, $\mathbf{Z} = \{\ldots,-1,0,1,\ldots\}$, and consider the set of samples inside a $2N+1 \times 2N+1$ window centered at the (n_1,n_2) location. Define the subsets, $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{W_4}$, of the square window as

$$\begin{aligned} \mathbf{W}_1[a;(n_1,n_2)] &= \{a(n_1+l_1,n_2) \ : \ -N \le l_1 \le N\}, \\ \mathbf{W}_2[a;(n_1,n_2)] &= \{a(n_1+l_1,n_2+l_1) \ : \ -N \le l_1 \le N\}, \\ \mathbf{W}_3[a;(n_1,n_2)] &= \{a(n_1,n_2+l_1) \ : \ -N \le l_1 \le N\}, \\ \mathbf{W}_4[a;(n_1,n_2)] &= \{a(n_1+l_1,n_2-l_1) \ : \ -N \le l_1 \le N\}. \end{aligned} \tag{1}$$

These sets specify the samples inside the masks shown in Fig.1a. Moreover, let

$$z_l(n_1,n_2) = median\Big[a(.,.) \in \mathbf{W}_l[a;(n_1,n_2)]\Big] \qquad 1 \le l \le 4. \tag{2}$$

Filters using the windows in (1) are unidirectional because the subwindows, $\mathbf{W}_l$, span one direction only. Using (2), the following unidirectional filters are defined:

Definition 1 : The output of the *Multilevel Max/median* filter is defined by

$$y_{m/max}(n_1,n_2) = med\Big[y_{max}(n_1,n_2),\ y_{min}(n_1,n_2),\ a(n_1,n_2)\Big], \tag{3}$$

where

$$\begin{aligned} y_{max}(n_1,n_2) &= max_{1\le l\le 4}\Big[z_l(n_1,n_2)\Big], \\ y_{min}(n_1,n_2) &= min_{1\le l\le 4}\Big[z_l(n_1,n_2)\Big]. \end{aligned} \tag{4}$$

□

We can see that the multilevel max/median filter is a combination of the max/median and min/median filters. The max/median (min/median) filter was shown to preserve positive (negative) image features, as well as to attenuate noise effectively [2]. In applications where both positive and negative features exist, the max/median or min/median filter, alone, are not adequate. Another filter structure that uses combinations of medians was proposed in [1] and is defined next.

Definition 2 : The *Multilevel Median* filter output is described by

$$y_{m/med}(n_1,n_2) = med\Big[y_{w_1,w_3}(n_1,n_2),\ y_{w_2,w_4}(n_1,n_2),\ a(n_1,n_2)\Big], \tag{5}$$

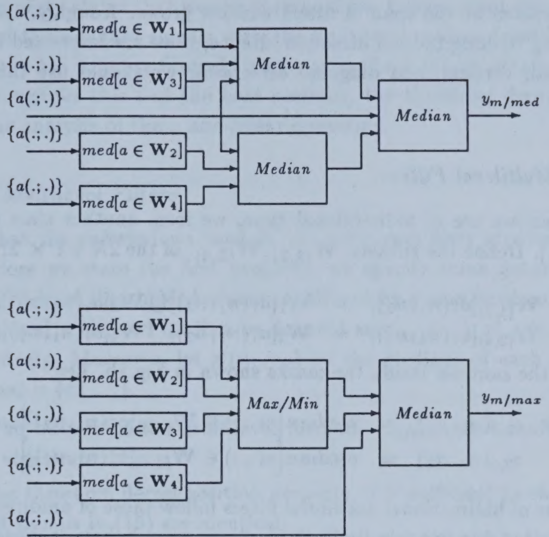


Figure 2: (a) Multilevel Median Filter; (b) Multilevel Max/Median.

where

$$\begin{aligned} y_{w_1,w_3}(n_1,n_2) &= median\Big[z_1(n_1,n_2), z_3(n_1,n_2), a(n_1,n_2)\Big], \\ y_{w_2,w_4}(n_1,n_2) &= median\Big[z_2(n_1,n_2), z_4(n_1,n_2), a(n_1,n_2)\Big]. \end{aligned} \tag{6}$$

□

By replacing the median within the z_l structures, by an averaging operation we form the class of *FIR-median Hybrid* filters[1].

Definition 3 : The *FIR-median Hybrid* filter output is described by

$$y_{fir}(n_1,n_2) = med\Big[\bar{y}_{w_1,w_3}(n_1,n_2),\ \bar{y}_{w_2,w_4}(n_1,n_2),\ a(n_1,n_2)\Big], \tag{7}$$

where

$$\begin{aligned} \bar{y}_{w_1,w_3}(n_1,n_2) &= median\Big[\hat{z}_1(n_1,n_2), \hat{z}_3(n_1,n_2), a(n_1,n_2)\Big], \\ \bar{y}_{w_2,w_4}(n_1,n_2) &= median\Big[\hat{z}_2(n_1,n_2), \hat{z}_4(n_1,n_2), a(n_1,n_2)\Big], \end{aligned} \tag{8}$$

where

$$\hat{z}_l = med\Big[z_{la}(n_1,n_2),\ z_{lb}(n_1,n_2),\ a(n_1,n_2)\Big], \qquad 1 \leq l \leq 4, \tag{9}$$

and where z_{la} and z_{lb} are the average of the samples in each of the two halfs of the window $\mathbf{W}_l$, excluding the center sample $a(n_1,n_2)$. □

In [1], the unidirectional multilevel median and FIR-median hybrid filters are referred to as the 3-level median (3LM-) and the 3-level unidirectional FIR-median hybrid filters (3LH-), respectively. These filters are denoted as "3-level" due to the three sequential operations used by these two multilevel filters. It should be mentioned that, for all filters considered here, we have limited the number of unidirectional window subsets, $\mathbf{W}_l$, to four. The number of window subsets can be increased as the span of filters window grows. Roughly, each substructure will preserve detail along its longitudinal direction. Hence, if we are interested in preserving details along the horizontal, vertical, and diagonal directions, we should use the four substructures defined above.

B. Bidirectional Multilevel Filters

If the basic subwindows are of bidirectional type, we form another class of multilevel detail-preserving filters [1]. Define the subsets, $\mathbf{W}_{(1,3)}, \mathbf{W}_{(2,4)}$, of the $2N+1 \times 2N+1$ square window as

$$\begin{aligned} \mathbf{W}_{(1,3)}[a;(n_1,n_2)] &= \mathbf{W}_1[a;(n_1,n_2)] \bigcup \mathbf{W}_3[a;(n_1,n_2)], \\ \mathbf{W}_{(2,4)}[a;(n_1,n_2)] &= \mathbf{W}_2[a;(n_1,n_2)] \bigcup \mathbf{W}_4[a;(n_1,n_2)]. \end{aligned} \tag{10}$$

These sets specify the samples inside the masks shown in Fig.1b. Let

$$\begin{aligned} z_{1,3}(n_1,n_2) &= median\Big[a(.,.) \in \mathbf{W}_{(1,3)}[a;(n_1,n_2)]\Big], \\ z_{2,4}(n_1,n_2) &= median\Big[a(.,.) \in \mathbf{W}_{(2,4)}[a;(n_1,n_2)]\Big], \end{aligned} \tag{11}$$

then, the definitions of bidirectional multilevel filters follow those of unidirectional filters. These definitions are ommited due to space limitations. Since we only have 2 subsets of in the window, as seen in Fig. 1a, we only need 2-levels of processing. The bidirectional multilevel median and FIR-median hybrid filters are denoted as the two-level median (2LM+) and the 2-level FIR-median hybrid (2LH+) filters, respectively.

2 Multilevel Median and Max/Median Filters

In this section, we will derive some deterministic properties of this filter. In particular, we show that both the multilevel median and multilevel max/median filters have the *Threshold decomposition* property [11]; furthermore, we show that these two filters are identical.

Let the input to the multilevel filters be $\{a(n_1,n_2)\}$. The thresholded signal $\{t^i(n_1,n_2)\}$ associated with the input is defined by

$$t^i(n_1,n_2) = \begin{cases} 1 & \text{if } a(n_1,n_2) \geq i; \\ 0 & \text{if } a(n_1,n_2) < i, \end{cases} \tag{12}$$

where $1 \leq i \leq k-1$ is the level of thresholding. Since the multilevel median and multilevel max/median operations embodies positive Boolean functions only, it follows from the theory of stack filters that these multilevel filters have the threshold decomposition attribute [12]. Hence, if we denote either of the multilevel median operators (unidirectional or bidirectional) on the input signal by $m/med[a(.,.)]$, and either of the multilevel max/median operator by $m/max[a(.,.)]$, then

$$y_{m/max}(n_1,n_2) = m/max\Big[a(n_1,n_2)\Big] = \sum_{i=1}^{k-1} x^i_{m/max}(n_1,n_2), \tag{13}$$

and

$$y_{m/med}(n_1,n_2) = m/med\Big[a(n_1,n_2)\Big] = \sum_{i=1}^{k-1} x^i_{m/med}(n_1,n_2), \tag{14}$$

where

$$\begin{aligned} x^i_{m/max}(n_1,n_2) &= m/max\Big[t^i(n_1,n_2)\Big], \\ x^i_{m/med}(n_1,n_2) &= m/med\Big[t^i(n_1,n_2)\Big]. \end{aligned} \tag{15}$$

In summary, this property states that we can threshold the k-level input signal into $k-1$ binary signals, filter each of these binary signals with the multilevel median (multilevel max/median) filter, and combine the $k-1$ binary filtered signals, as in (14)-(15), to obtain the k-level output. As will be seen throughout this and the next sections, the threshold decomposition property greatly simplifies the analysis of these non-linear operators.

A. Unidirectional Multilevel Filters

Here, we assume that the subwindows used in the multilevel filter structures are of a unidirectional type. Before we state the first property, we specify some notation for the sake of simplicity. Let the $i'th$-level thresholded sequence, filtered by a max/median, min/median, multilevel max/median, and multilevel median be denoted as $x^i_{max}(.,.)$, $x^i_{min}(.,.)$, $x^i_{m/max}(.,.)$ and $x^i_{m/med}(.,.)$, respectively. Moreover, let $z^i_l(n_1,n_2)$ be the medians of each substructure in (2) when the input signal is $\{t^i(.,.)\}$.

Property 1 : The output of the multilevel median, $y_{m/med}$, and multilevel max/median filters, $y_{m/max}$, are identical.

Proof : Using the threshold decomposition property, it is sufficient to show that each of the $k-1$ binary filtered signals in (15) are identical.

Case 1: Assume $t^i(n_1,n_2)=0$. With this condition the output of the max/median can be expressed as $x^i_{m/max}(n_1,n_2)=med[x^i_{max}(n_1,n_2),x^i_{min}(n_1,n_2),0]$. Since $x^i_{min}\le x^i_{max}$, then

$$\begin{aligned} x^i_{m/max}(n_1,n_2) &= 1-I(x^i_{min}(n_1,n_2)=0), \\ &= 1-I(\text{at least one subfilter median}, z^i_l, \text{ is zero}), \end{aligned} \tag{16}$$

where $I(A)$ is the indicator function

$$I(A)=\begin{cases} 1, & \text{if } A \text{ is true;} \\ 0, & \text{if } A \text{ is false.} \end{cases} \tag{24}$$

Referring to (5), the output of the multilevel median is

$$x^i_{m/med}(n_1,n_2)=med[x^i_{w_1w_3}(n_1,n_2),x^i_{w_2w_4}(n_1,n_2),0]. \tag{18}$$

If any of the subfilter medians z^i_l is zero, then at least one term inside the median operator in (18) is also zero, consequently

$$x^i_{m/med}(n_1,n_2)=1-I(\text{at least one subfilter median}, z^i_l, \text{ is zero}). \tag{19}$$

Since (20) and (19) are identical, so are the outputs of these filters.

Case 2: Assume $t^i(n_1,n_2)=1$. With this condition the output of the max/median can be expressed as $x^i_{m/max}(n_1,n_2)=med[x^i_{max}(n_1,n_2),x^i_{min}(n_1,n_2),1]$. Since $x^i_{min}\le x^i_{max}$, then

$$\begin{aligned} x^i_{m/max}(n_1,n_2) &= 1-I(x^i_{max}(n_1,n_2)=0) \\ &= 1-I(\text{all of the subfilter medians}, z^i_l, \text{ are zero}). \end{aligned} \tag{20}$$

Similarly, the output of the multilevel median is

$$x^i_{m/med}(n_1,n_2)=med[x^i_{w_1w_3}(n_1,n_2),x^i_{w_2w_4}(n_1,n_2),1]. \tag{21}$$

If any of the subfilter medians z^i_l is one, then at least one of the two terms inside the median operator in (21) is one; thus,

$$\begin{aligned} x^i_{m/med}(n_1,n_2) &= I(\text{at least one subfilter median}, z^i_l, \text{ is one}), \\ &= 1-I(\text{all of the subfilter medians}, z^i_l, \text{ are zero}). \end{aligned} \tag{22}$$

Since (22) is identical to (20), property 1 is proved. □

B. Bidirectional Multilevel Filters

Let the $z^i_{1,3}(n_1, n_2)$ and $z^i_{2,4}(n_1, n_2)$ be the medians of each substructure in (10) when the input signal is $\{t^i(.,.)\}$. Since the samples in the two bidirectional subwindows, $\mathbf{W}_{(1,3)}$ and $\mathbf{W}_{(2,4)}$, are mutually exclusive except for the center sample, the proof of the next property follows closely the proof of property 1.

Property 2 : The output of the bidirectional multilevel median, $y_{m/med}$, and multilevel max/median filters, $y_{m/max}$, are identical.

3 Statistical Threshold Decomposition For Multilevel Median Filters

The smoothing performance of the multilevel median filter can be described by the multivariate output distribution. In general, this is not possible since accurate statistical descriptions of input images are difficult to obtain. It is possible, however, to describe the noise attenuation characteristics of multilevel-median filters by the probability measure function (pmf) of the filtered signals in homogeneous regions.

Statistical Threshold Decomposition was introduced as a method to describe marginal probability measure functions of median filtered sequences, using probability measure functions of binary median filtered sequences[11]. Here we extend those concepts to the multilevel-median filtering of two-dimensional sequences.

Let $\{a(n_1, n_2) : n_1, n_2 \in \mathbf{Z};\}$ be an independent, identically distributed (i.i.d.), discrete alphabet random sequence, with a probability space $(\mathbf{\Omega}, \mathbf{F}, \mathbf{P})$, where the sample space is $\mathbf{\Omega} = \mathbf{Z}_k = \{0, 1, \ldots, k-1\}$. The event space $\mathbf{F}$ is the power set, and $\mathbf{P}$ is the probability measure described by the discrete distribution function $F(i), i \in \mathbf{Z}_k$. The thresholded binary sequence $\{t^i(n_1, n_2)\}$ is also i.i.d., with a probability space $(\mathbf{\Omega_b}, \mathbf{F_b}, \mathbf{P_b})$, where the sample space is $\mathbf{\Omega_b} = \{0, 1\}, \mathbf{F_b}$ is the binary power set, and $\mathbf{P}_b$ induced by the probability measure function (pmf)

$$\begin{aligned} Pr(t^i(n_1, n_2) = 0) &= F(i-1), \\ Pr(t^i(n_1, n_2) = 1) &= 1 - F(i-1). \end{aligned} \tag{23}$$

A. Unidirectional Windows

Refer to the substructure medians at the i'th threshold level as

$$z^i_l(n_1, n_2) = median\left[t^i(.,.) \in \mathbf{W}_l[t^i; (n_1, n_2)]\right], \quad l = 1, \ldots 4. \tag{24}$$

The max/median and min/median at the binary level are denoted as $x^i_{max}(n_1, n_2)$ and $x^i_{min}(n_1, n_2)$, respectively. The multilevel median filter at the $i'th$ level is denoted as $x^i_{m/med}(n_1, n_2) = x^i_{m/max}(n_1, n_2)$. With this notation, the distribution of the unidirectional multilevel median filter follows.

Property 3 : If $\{y_{m/med}(n_1, n_2); y \in \mathbf{Z}_k\}$ and $\{x^i_{m/med}(n_1, n_2); x^i \in (0, 1); 1 \leq i \leq k-1\}$ are the k-level, and thresholded multilevel median filtered signals, then

$$\begin{aligned} F_y(j) &= P_r\left[y_{m/med}(n_1, n_2) \leq j\right] = P_r\left[x^{j+1}_{m/med}(n_1, n_2) = 0\right] \\ &= F(j)\left[1 - \left(\textstyle\sum_{q=N+1}^{2N} \binom{2N}{q}(1 - F(j))^q F(j)^{2N-q}\right)^4\right] \\ &+ (1 - F(j))\left[\textstyle\sum_{q=N+1}^{2N} \binom{2N}{q}(1 - F(j))^{2N-q} F(j)^q\right]^4, \end{aligned} \tag{25}$$

where $F(j)$ is the probability measure function of the i.i.d. input sequence $\{a(.,.)\}$.

Proof: Using the total law of probability, we obtain

$$\begin{aligned} P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\right] = {} & P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right]P_r\left[t^{j+1}(n_1,n_2)=0\right] \\ & + P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=1\right]P_r\left[t^{j+1}(n_1,n_2)=1\right]. \end{aligned} \tag{26}$$

Applying the definition of multilevel median filters on the first term of (31), we obtain

$$P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right] = P_r\left[med[x^{j+1}_{max}(n_1,n_2),x^{j+1}_{min}(n_1,n_2),0]=0\right]. \tag{27}$$

Since $x^{j+1}_{min} \leq x^{j+1}_{max}$i, the above equation reduces to

$$\begin{aligned} P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right] & = P_r\left[x^{j+1}_{min}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right] \\ & = P_r\left[\min_{1\leq l\leq 4}\left(z^{j+1}_l(n_1,n_2)\right)=0\big/t^{j+1}(n_1,n_2)=0\right] \\ & = P_r\left[\text{at least one } z^{j+1}_l(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right] \\ & = 1-P_r\left[\text{all } \; z^{j+1}_l(n_1,n_2)=1\big/t^{j+1}(n_1,n_2)=0\right]. \end{aligned} \tag{28}$$

Since

$$\begin{aligned} P_r\left[z^{j+1}_l(n_1,n_2)=1\big/t^{j+1}(n_1,n_2)=0\right] & = P_r\left[\text{at least } N+1 \text{ samples are one}\big/t^{j+1}(n_1,n_2)=0\right] \\ & = \textstyle\sum_{q=N+1}^{2N}\binom{2N}{q}(1-F(j))^q F(j)^{2N-q}, \end{aligned} \tag{29}$$

equation (28) reduces to

$$P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=0\right] = \left[1-\Big(\sum_{q=N+1}^{2N}\binom{2N}{q}(1-F(j))^q F(j)^{2N-q}\Big)^4\right]. \tag{30}$$

Similarly, applying the definition of multilevel median filters on the second term in (31), and following the same procedure as in (32)-(35), we obtain

$$P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\big/t^{j+1}(n_1,n_2)=1\right] = \left[1-\Big(\sum_{q=N+1}^{2N}\binom{2N}{q}(1-F(j))^{2N-q} F(j)^{q}\Big)^4\right]. \tag{31}$$

Property 3 follows directly from (26), (30), and (31). □

B. Bidirectional Windows

Each bidirectional subwindow spans $4N+1$ samples, as opposed to $2N+1$ samples spanned by the unidirectional subwindow. Thus for the same $2N+1 \times 2N+1$ window span, the bidirectional filter will have better noise removal, and worse detail-preserving characteristics than the unidirectional multilevel median filter. The output distributions quantify the noise attenuation improvement.

Property 4 : If $\{y_{m/med}(n_1,n_2); y \in \mathbf{Z}_k\}$ and $\{x^i_{m/med}(n_1,n_2); x^i \in (0,1); 1 \leq i \leq k-1\}$ are the k-level, and thresholded bidirectional, multilevel median filtered signals, then

$$\begin{aligned} F_y(j) & = P_r\left[y_{m/med}(n_1,n_2)\leq j\right] = P_r\left[x^{j+1}_{m/med}(n_1,n_2)=0\right] \\ & = F(j)\left[1-\Big(\textstyle\sum_{q=2N+1}^{4N}\binom{4N}{q}(1-F(j))^q F(j)^{4N-q}\Big)^2\right] \\ & + (1-F(j))\left[\textstyle\sum_{q=2N+1}^{4N}\binom{4N}{q}(1-F(j))^{4N-q} F(j)^q\right]^2, \end{aligned} \tag{32}$$

where $F(j)$ is the probability measure function of the i.i.d. input sequence $\{a(.,.)\}$.

Breakdown Probability For *3Level-Median Filter*						
Probability, p	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$
0.0625	0.025205	0.005211	0.001051	0.000217	0.000046	0.000010
0.125	0.082049	0.035005	0.013944	0.005530	0.002212	0.000895
0.25	0.224983	0.175733	0.131032	0.095817	0.694390	0.050122
0.375	0.366513	0.348164	0.327956	0.307718	0.288039	0.269162
0.50	0.50	0.50	0.50	0.50	0.50	0.50

Table 1: **Breakdown probabilities for 3Level-Median Filter**

Breakdown Probability For *2Level Bidirectional Median Filter*						
Probability, p	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$
0.0625	0.002663	0.000109	0.000005	2×10^{-7}	1×10^{-7}	6×10^{-10}
0.125	0.018980	0.002797	0.000448	0.000075	0.000013	0.000002
0.25	0.115669	0.054228	0.026613	0.013422	0.006896	0.003590
0.375	0.288434	0.227696	0.184333	0.151405	0.125558	0.104846
0.50	0.50	0.50	0.50	0.50	0.50	0.50

Table 2: **Breakdown probabilities for 2Level Bidirectional Median Filter**

C. Breakdown Probabilities

In impulsive noise environments, if there are several noisy samples clustered together, impulses may not be removed from the image. It is very helpful to quantify the probability of such an event. Define the *Breakdown Probability* as the probability of an impulse occurring at the output of the multilevel median filter, and let p be the occurrence probability of an impulse at the input of the filter (the impulse is equally likely to be positive or negative). The breakdown probabilities, in homogeneous image regions, follows from properties 3 and 4.

Property 5 : Let p be the probability of an impulse occurring at the input, then the breakdown probabilities for the unidirectional, $P_r\{B_N^-\}$, and the bidirectional, $P_r\{B_N^+\}$, multilevel median filters are:

$$P_r\Big[B_N^-\Big] = p\left[1 - \Big(\sum_{q=N+1}^{2N} \binom{2N}{q}(1-p)^q p^{2N-q}\Big)^4\right] + (1-p)\left[\sum_{q=N+1}^{2N} \binom{2N}{q}(1-p)^{2N-q} p^q\right]^4, \quad (33)$$

and

$$P_r\Big[B_N^+\Big] = p\left[1 - \Big(\sum_{q=2N+1}^{4N} \binom{4N}{q}(1-p)^q p^{4N-q}\Big)^2\right] + (1-p)\left[\sum_{q=2N+1}^{4N} \binom{4N}{q}(1-p)^{4N-q} p^q\right]^2, \quad (34)$$

respectively. □

Tables 1 and 2 show the breakdown probabilities for different values of p, and N; hence, for a given performance we can easily select the size of the filter. Clearly, the bidirectional multilevel median filters have better noise attenuation characteristics at the expense of having less detail-preserving properties.

To illustrate the filtering performance of each filter, Fig. 3 shows a) the original image, b) the noisy image (impulsive), c) the square median filter, d) the multilevel hybrid FIR-median (3LH-), e) the unidirectional multilevel median (3LM-), and d) the bidirectional multilevel median (2LM+) filtered images, respectively.

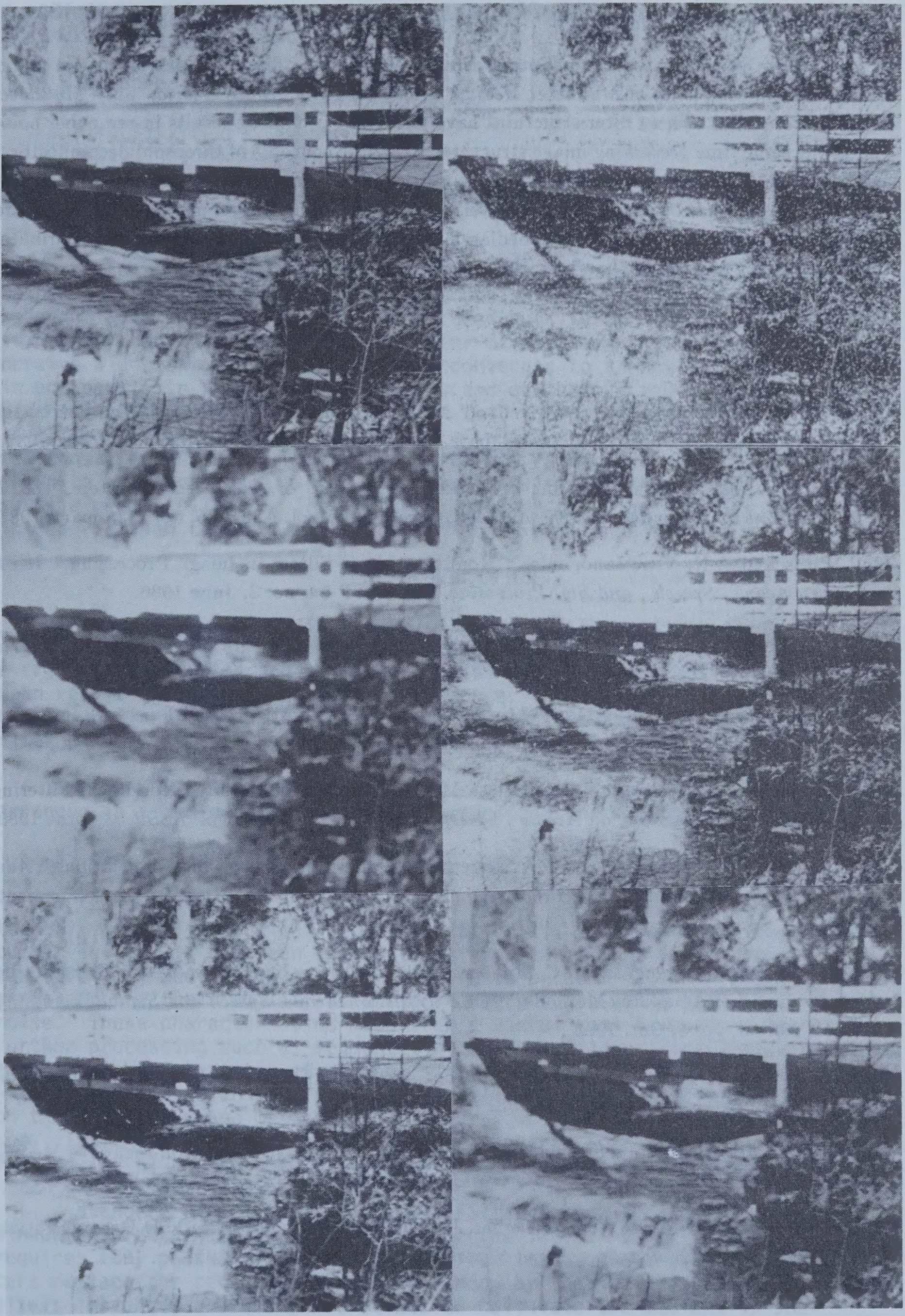


Figure 3: a) original image, b) noisy image (impulsive), c) square median filter, d) multilevel hybrid FIR-median (3LH-), e) multilevel median (3LM-), and d) bidirectional multilevel median (2LM+) filtered images, respectively.

4 Conclusions

In this paper, we considered non-adaptive, structure preserving filters. In particular, we found the theoretical properties of *Multilevel Median, Multilevel FIR-Median Hybrid,* and *Multilevel Max/Median* filters. These filter structures have yield very promising results in removing noise and at the same time preserving image structure. Using the concepts of threshold decomposition and statistical threshold decomposition, we derived closed form expression for output distribution functions. It is shown that the multilevel median and the multilevel max/median filters are identical. Since the class of multilevel median filters have the threshold decomposition attribute, they are simpler to analyze than their FIR-median hybrid counterparts.

References

[1] A. Nieminen, P. Heinonen, and Y. Neuvo," A new class of Detail-Preserving Filters for Image Processing," *IEEE Trans. on Patt. Analysis and Mach. Int.,* Vol. PAMI-9, no. 1, Jan. 1987.

[2] G.R. Arce and M.P. Mcloughlin, " Theoretical Analysis of Max/Median Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. ASSP-35, no. 1, January, 1987.

[3] I. Pitas and A.N. Venetsanopoulos," Nonlinear Mean Filters in Image Processing," *IEEE Trans., Acous., Speech , and Sig. Processing,* Vol. ASSP-34, no. 3, June 1986.

[4] Y.H. Lee and A.T. Fam, "An Edge Gradient Enhancing Adaptive Order Statistic Filter", *IEEE Trans. on Acous. , Speech, and Signal Processing,* Vol. ASSP-35, No.5, May 1987.

[5] R.L. Stevenson and G.R. Arce , " Morphological Filters: Statistics and Further Syntactical Properties," *IEEE Trans. on Circuits and Systems.* Vol. CAS-34, No. 11, Nov. 1987.

[6] Y.H. Lee and S.A. Kassam, " Generalized Median Filtering and Related Non-linear Filtering Techniques," *IEEE Trans. on Acous. , Speech, and Sig. Processing,* Vol. ASSP-33, June 1985.

[7] G.R. Arce, N.C. Gallagher, and T.A. Nodes, " Median Filter: Theory and Applications," *Adv. in Computer Vision and Image Processing* , T.S. Huang, Ed. , Vol. 2 JAI Press, Connecticut, 1986.

[8] A.C. Bovik, T.S. Huang, and D.C. Munson, " Edge-Sensitive Image Restoration Using Order-Constrained Least Squares Methods," *IEEE Trans. on Acous. Speech and Signal Processing,* Vol. ASSP-33, no. 5, Oct. 1985 .

[9] P.D. Wendt, E.J. Coyle, and N.C. Gallagher, Jr."Stack Filters, " *IEEE Trans. on Acous. Speech and Signal Processing,* Vol. ASSP-34, Aug. 1986.

[10] G.R. Arce, "Statistical Threshold Decomposition for Recursive and Non-recursive Median Filters, " *IEEE Transactions on Information Theory,* Vol.IT-32, No. 2, March 1986.

[11] P. Fitch, E.J. Coyle, and N.C. Gallagher, Jr."Median Filtering by Threshold Decomposition, " *IEEE Trans. on Acous. Speech and Signal Processing,* Vol. ASSP-32, Dec. 1984.

A FAST RECURSIVE TWO-DIMENSIONAL MEDIAN FILTERING ALGORITHM FOR MULTI-LEVEL IMAGE ROOT EXTRACTION

W.W. Boles, M. Kanefsky and M. Simaan
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261

ABSTRACT

Repeated applications of two-dimensional median filters over an image converge to an invariant signal called a root. Root signals can be obtained using either a standard or a recursive median filter. The convergence rate of a two-dimensional recursive filter to a root of an image is higher than that of a standard filter of the same size. Our first recursive algorithm, developed earlier, converges to a root of a binary image in one pass. In this paper we develop and evaluate a new recursive two-dimensional median filtering algorithm that converges to a root of a multi-level image in two passes, the second of which includes a number of local re-scans. The number of ordering operations required by our algorithm is shown to be significantly less than that required by the standard and recursive median filters. The roots achieved by our algorithm are different in general than those of either of the above mentioned filters.

I. INTRODUCTION

In many signal processing applications, the suppression of noise in a digitized image is achieved by linear estimators. Linear estimation, however, is greatly affected by impulse noise or sudden changes in the image mean resulting in an inadequate performance. It is in such cases that robust nonlinear estimators using order statistics appear to be a possible alternative.

If the $K^2=2L+1$ pixels in a KxK square window (where $K=2N+1$) are arranged in a non-decreasing order of magnitude and written as:

$$X_{(1)} \leq X_{(2)} \leq X_{(3)} \leq \cdots \leq X_{(2L+1)} \tag{1}$$

then $X_{(i)}$ is called the i^{th} order statistic [1]. The median filter whose output is $X_{(L+1)}$, has been successfully applied in many image processing tasks [2-4]. Median filters are mainly used because of their easy implementation and their ability to preserve sharp edges while removing impulse noise. These characteristics make them a useful tool to prepare images for further processing such as segmentation or region identification. Another important characteristic of median filters is that - when used iteratively - they converge to an invariant signal called a root. Root signals depend on the size and shape of the window used as well as the type of median filter. They have been studied for binary images for one-dimensional median filters [5, 6] and to some extent for two-dimensional filters [2].

A standard two-dimensional median filter typically uses a square window of size $K^2=2L+1$ which passes over the image. Convergence to a root requires many passes of the filter over the image. Recursive median filters replace the center pixel of the window by the median value of all the pixels inside it and use these values to calculate the medians in subsequent window positions. Convergence to a root of a one-dimensional signal requires only one pass of the recursive median filter, however, this is not generally the case for two-dimensional signals [7].

II. CONVERGENCE RATES OF MEDIAN FILTERS

When a median filter's window is passed over an image, the median values could be different from the input pixel values occupying the center of the window. These situations will be referred to simply as "pixel changes." When a median filter is used repeatedly to produce a root, the number of pixel changes will be at its peak at the first pass of the filter over the image. The number of pixel changes drops dramatically for the next few passes. As the resulting outputs of successive passes approach the root, the number of pixel changes becomes smaller and smaller. To illustrate this characteristic, consider the image shown in Figure (1). The convergence rates of a 3x3 standard and recursive median filters are shown in Figure (2). In such situations, many ordering operations are performed for very little change in the final output. Although the number of passes required by the regular two-dimensional recursive median filter to converge to a root of an image is less than that required by a standard filter of the same size, it still requires a considerable number of passes over the image.

III. A FAST RECURSIVE ALGORITHM FOR MULTI-LEVEL IMAGES

A recursive median filter uses in each window position L pixels which have been already scanned to calculate the output. If a pixel change occurs during a filter's pass, the new pixel value will (by construction) be used to calculate the medians of subsequent window positions. Also, that same change may affect some of the previously scanned pixel values. When this happens, it will require at least one more pass over the image to converge to a root. Our new algorithm determines the exact previously scanned area of the image affected by a single pixel change. This area consists of the center pixels of all window positions that used the current pixel in the calculation of their respective medians. These center pixels are actually the L previously scanned pixels of the current window position. Assume that the window moves left to right in each row and top to bottom for row advances. Our initial algorithm, developed mainly for binary images [8], identifies the affected area to be N columns to the left of the center pixel of the current window position and N rows above it. It then backtracks to the beginning of that area and re-scans the pixels inside it. If in the process other pixels change value, the algorithm backtracks again. In this way a root is reached in only one complete pass which includes a number of local re-scans.

For multi-level images, the number of local re-scans could be quite large, affecting the speed of convergence to a root. Thus we modify our algorithm for multi-level images in such a way as to converge to a root in two passes instead of one. The first pass of our algorithm corresponds to the regular recursive median filter. Referring to the discussion presented in the previous section, the first pass of a recursive median filter achieves the peak number of pixel changes toward a root. The second pass monitors the possible pixel changes to re-scan the respective affected areas in the same way our first algorithm did. In this sense, this modified algorithm converges to a root of a multi-level image even faster than our earlier one [8]. A flow chart for this algorithm is shown in Figure (3). We note at this stage that the roots obtained by our algorithms are different in general than those of either the standard or recursive filters.

IV. PERFORMANCE ANALYSIS OF THE ALGORITHM

For a certain image, if a root is obtained using a standard median filter after P1 passes, then a regular recursive median filter of the same size will converge to a root after P2 passes where P2<P1. The number of ordering operations, Ri, required to converge to a root of an image of size M1xM2, using a square window of size $K^2=2L+1$ and a median filter is given by:

$$Ri = Pi * (M1 * M2) * Q(L) \tag{2}$$

where i=1,2 refers to the standard and recursive median filters respectively and Q(L) is the number of ordering operations for each window of size 2L+1.

Our first algorithm converges to a root in one pass and requires a number of ordering operations, R3, which is given by [8]:

$$R3 = (P_\ell * (L + 1) + (M1 * M2)) * Q(L) \tag{3}$$

where P_ℓ is the number of local re-scans.

Dividing equations (2) and (3) by M1xM2, and the number Q(L) which are common factors, the convergence rates of the standard, recursive and new filters can be compared based on the number of passes P1, P2 and the effective number of passes, P3, for the first algorithm which is given by:

$$P3 = (P_\ell * (L + 1)/(M1 * M2)) + 1 \tag{4}$$

Expression (4) shows that even if the number of local re-scans, P_ℓ, of our first algorithm is much larger than P1 and P2, its effect will be dramatically reduced by the size of the image.

Since the number of ordering operations required by one pass of a recursive median filter is given by equation (2) with P2=1, then the total number of ordering operations, R4, required by our modified algorithm will be given by:

$$R4 = (\hat{P}_\ell * (L + 1) + (2*(M1 * M2))) * Q(L) \tag{5}$$

where $\hat{P}_\ell << P_\ell$.

The effective number of passes in this case, P4, is given by:

$$P4 = (\hat{P}_\ell * (L + 1)/(M1 * M2)) + 2 \tag{6}$$

which achieves further reductions in the effective number of passes required for convergence to a root of a multi-level image.

It is worth mentioning at this point that, for the standard or recursive median filters, an additional complete pass over the image is always necessary just to insure that a root has been achieved during the previous pass. Whenever either of our algorithms is used, there will be no need for such an extra pass, hence resulting in further computation savings.

V. EXAMPLE

Our algorithms have been tested for performance using the 80x80 (M1=M2=80), 256-level real image shown in Figure (1-a). A 3-D plot of the same image is shown in Figure (1-b). The viewing angles of this plot are chosen to best illustrate the performance of our algorithms. The standard and recursive filters as well as our algorithms were deployed to produce roots using a square 3x3 window. The standard filter converged to a root after P1=40 passes. The recursive filter required P2=20 passes to obtain a root. Our first algorithm [8] produced a root after one complete pass over the image including P_ℓ=7081 local re-scans with an effective number of passes, P3=6.53. For this example the algorithm performed 16.3% of the ordering operations required by the standard filter and 32.6% of those required by the recursive filter. For the same example, the modified algorithm required $\hat{P}_\ell$=2900 re-scans resulting in P4=4.27 effective passes. This represents only 10.7% and 21.4% as many operations as the standard and recursive filters respectively.

To show that the roots produced by our algorithms are different from those of either the standard or recursive filters, we used the 20x20 square segment on the upper left corner of the same image of Figure (1). This segment has only 16 gray levels and is shown in Figure (4). The roots produced by the standard, recursive, first and modified algorithms are shown in Figures (5-a to 5-d) respectively. Notice that the first algorithm produced a smoothed image segment while the modified algorithm converged even faster to a root that is similar to that of the recursive filter. In spite of the exaggerated differences presented above in the 20x20 piece of the image, the final roots of the entire 80x80 image for all four algorithms could not be easily distinguished from one another. For this reason, only the root of the multi-level algorithm is shown in Figure (6).

VI. CONCLUSIONS

A fast two-dimensional median filtering algorithm for image root extraction is presented. The performance of this algorithm showed substantial reductions in the number of ordering operations required to converge to a root of a multi-level image relative to the standard and recursive filters. The number of ordering operations required by this algorithm are also shown to be less than those required by an algorithm we previously developed for binary images. The roots achieved by our algorithms are slightly different in general than those obtained using the standard or recursive median filters which are also different. The algorithms may be used whenever fast root extraction is required for further use in subsequent image processing tasks.

REFERENCES

[1] H.A. David, Order Statistics, John Wiley, New York, 1981.

[2] T.S. Huang, Ed., Two-Dimensional Digital Signal Processing II, Transforms and Median Filters, Springer-Verlag, Berlin, 1981, Chapters 1, 5 and 6.

[3] G.R. Arce and N.C. Gallagher, "BTC Image Coding Using Median Filter Roots," IEEE Trans. Commun., COM-31, pp. 784-793, June 1983.

[4] R.M. Hodgson, D.G. Bailey, M.J. Naylor, A.L.M. Ng and S.J. McNeill, "Properties, Implementations and Applications of Rank Filters," Image and Vision Computing 1, No. 1, pp. 3-14, February 1985.

[5] P.D. Wendt, E.J. Coyle and N.C. Gallagher, "Some Convergence Properties and Median Filters," IEEE Trans. Acoust., Speech and Signal Processing, ASSP-33, pp. 276-286, March 1986.

[6] G.R. Arce and N.C. Gallagher, "State Description of the Root Signal Set of Median Filters," IEEE Trans. Acoust., Speech and Signal Processing, ASSP-30, pp. 894-902, December 1982.

[7] T.A. Nodes and N.C. Gallagher, "Median Filters: Some Modifications and Their Properties," IEEE Trans. Acoust., Speech and Signal Processing, ASSP-30, pp. 739-746, October 1982.

[8] W.W. Boles, M. Kanefsky and M. Simaan, "A Recursive Two-dimensional Median Filtering Algorithm," Proc. of IEEE International Symposium on Circuits and Systems, Philadelphia, PA, pp. 252-254, May 1987.

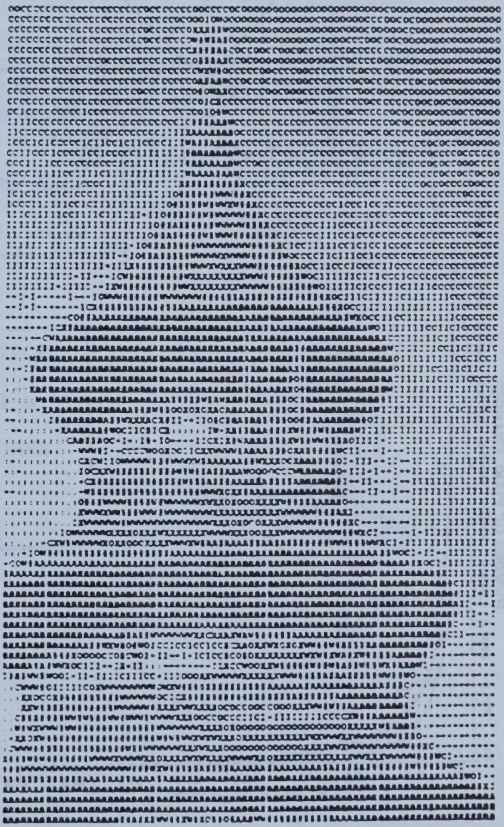


Figure (1-a) Test image (2-D).

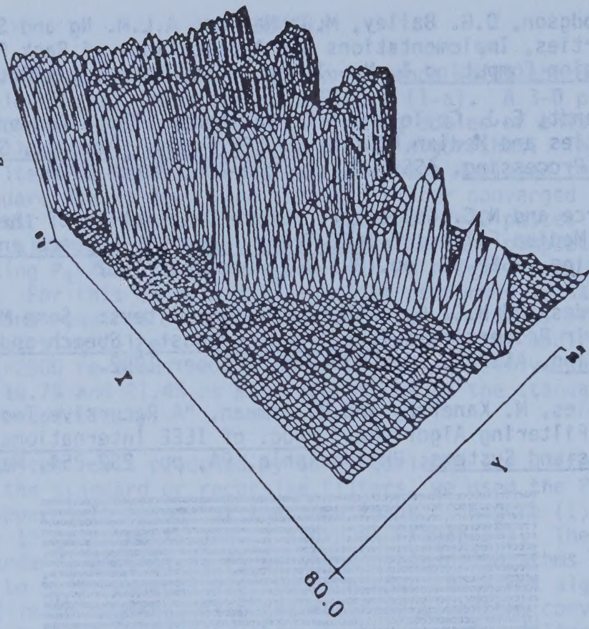

256.0
Z
Y
Y
80.0

Figure (1-b) Test image (3-D).

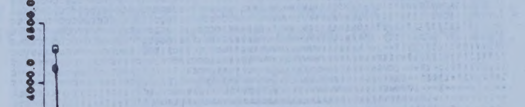


Figure (2) Convergence rates of the standard and recursive median filters

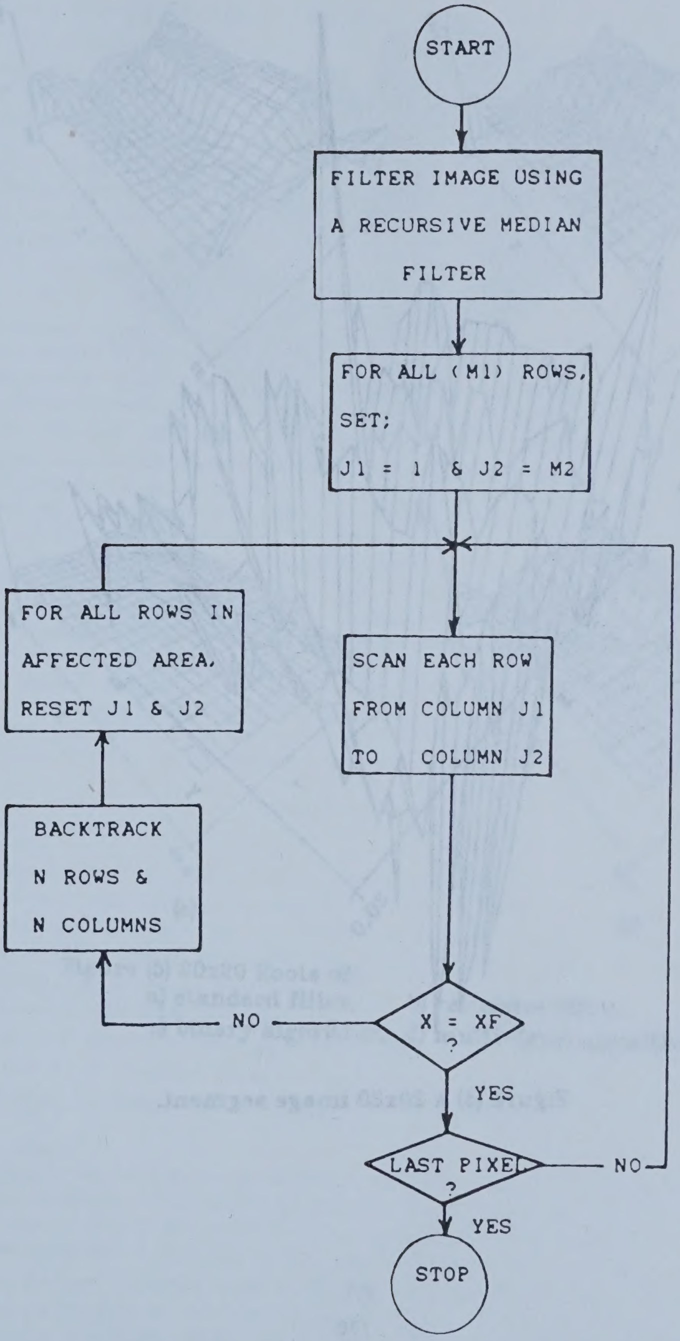

START
FILTER IMAGE USING A RECURSIVE MEDIAN FILTER
FOR ALL (M1) ROWS, SET; J1 = 1 & J2 = M2
FOR ALL ROWS IN AFFECTED AREA, RESET J1 & J2
SCAN EACH ROW FROM COLUMN J1 TO COLUMN J2
BACKTRACK N ROWS & N COLUMNS
NO
X = XF ?
YES
LAST PIXEL ?
NO
YES
STOP

FIGURE (3) FLOW CHART OF THE ALGORITHM

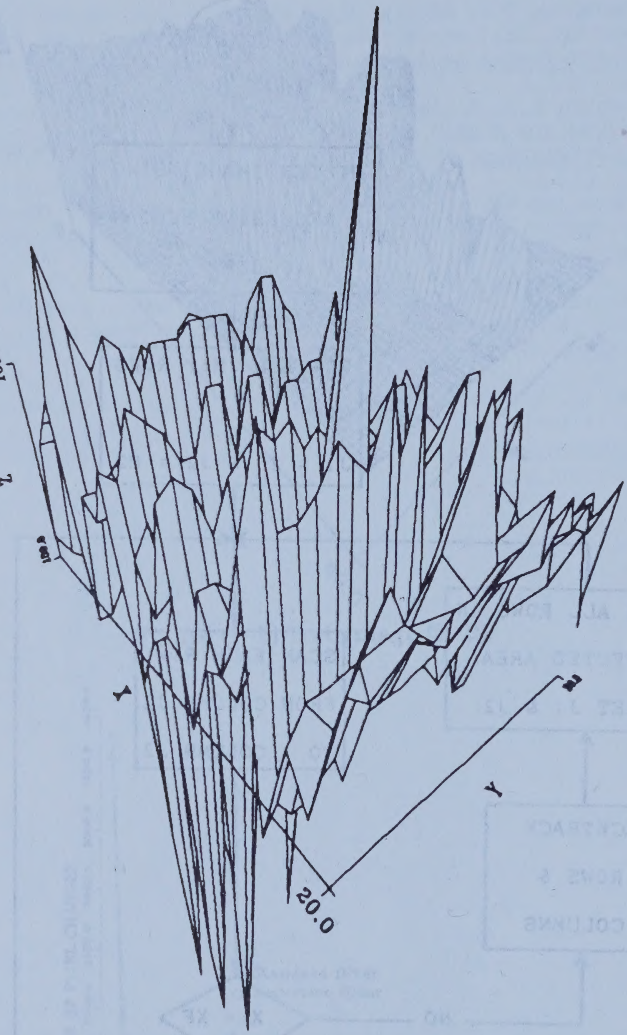
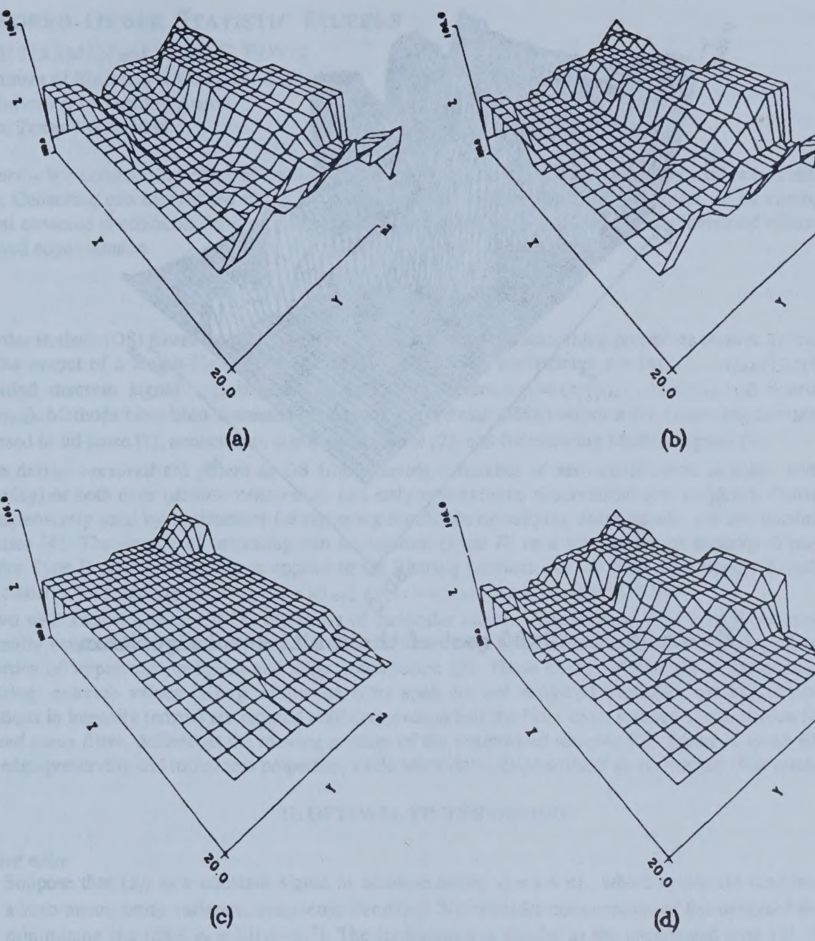


Figure (4) A 20x20 image segment.



(a) (b)

(c) (d)

Figure (5) 20x20 Roots of:
a) standard filter, b) recursive filter,
c) binary algorithm, d) multi-level algorithm.

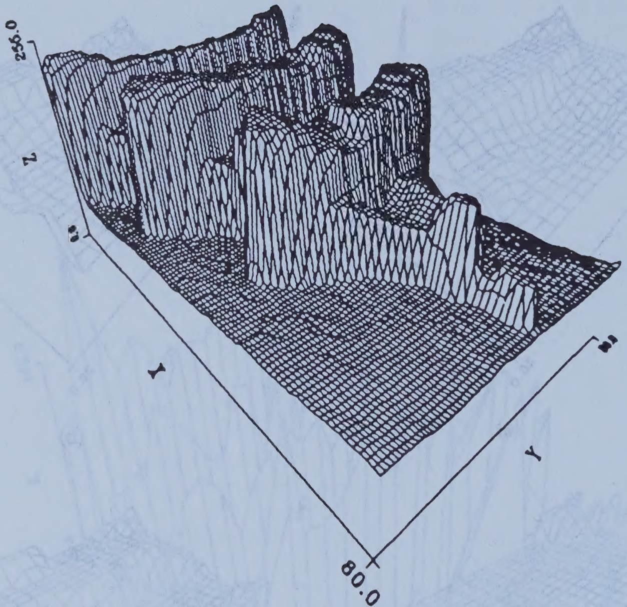


Figure (6) An 80x80 root of the multi-level algorithm.

CENSORED ORDER STATISTIC FILTERS

LAITH NAAMAN and ALAN C. BOVIK
Department of Electrical & Computer Engineering
The University of Texas at Austin
Austin, Texas 78712-1084

Abstract – We consider the effect of coefficient censoring on the smoothing performance of order statistic (OS) filters. Censoring can be effected either by adding zero coefficients to an existing filter, or by computing the optimal censored coefficients for a set filter span. In both cases, censoring often yields increased robustness and improved edge retention.

I. INTRODUCTION

Order statistic (OS) filters have been noted for their robust signal smoothing properties relative to linear filters [1]. The output of a length-$(2n+1)$ nonrecursive OS filter with coefficients $\mathbf{a} = (a_1, \ldots, a_{2n+1})$ applied to a two-sided discrete signal $\{x_i\}$ is given by $y_i = \mathbf{a} \cdot \mathbf{x}_{(i)}$, where $\mathbf{x}_{(i)} = (x_{(1):i}, \ldots, x_{(2n+1):i}) = \text{order}(x_{i-n}, \ldots, x_{i+n})$. Methods have been discussed for finding the optimal (MSE) vector $\mathbf{a}$ for estimating constant signals immersed in iid noise [1], nonconstant signals in iid noise [2], and for restoring Markov signals [3].

We define *censored OS filters* as OS filters having a number of zero coefficients at either end (*single censoring*) or both ends (*double censoring*), i.e., only non-extreme observations are weighted. Censoring has been extensively used by statisticians for removing unreliable or outlying observations, and for obtaining quick estimates [4]. The degree of censoring can be random (*Type I*), or a set number of samples (*Type II*). We consider Type II double censoring as applied to the filtering problem. An OS filter with exactly k $(0 \leq k \leq n)$ zero coefficients at each end $(a_1 = \ldots = a_k = 0 = a_{2n+2-k} = \ldots = a_{2n+1})$ is a k-censored OS filter.

Two well-known k-censored OS filters are of particular interest. The *median filter* is an n-censored (hence maximally censored) OS filter. The median filter is interesting since it possesses the unique signal smoothing properties of impulse suppression and edge preservation [5]. These characteristics are a consequence of the censoring: extreme values falling within the filter span are not weighted (impulses are suppressed), while transitions in intensity (edges) are ignored until they exceed half the filter span. Related to the median filter is the *trimmed mean filter*, defined as the moving average of the uncensored samples [6]. Trimmed mean filters have good edge-preserving and robustness properties, while not creating false artifacts as the median filter often does [7].

II. OPTIMAL FILTER DESIGN

Additive noise

Suppose that $\{x_i\}$ is a constant signal in additive noise: $x_i = s + n_i$, where n_i are iid random variates from a zero-mean, unity variance, symmetric density f. We consider computation of the unbiased k-censored filter minimizing the MSE $e_k = \text{E}[(y_i - s)^2]$. The formulation is similar to the uncensored case [1]. As in [1], the problem is cast as a quadratic optimization problem; we shall depend heavily on the formulation given there.

The output of a k-censored filter with input $\{x_i\}$ is $y_i = s(\mathbf{1} \cdot \mathbf{a}_k) + \mathbf{n}_{(i),k} \cdot \mathbf{a}_k$, where $\mathbf{n}_{(i),k}$ is the vector of censored, ordered noise samples, $\mathbf{1} = (1, \ldots, 1)$, and $\mathbf{a}_k$ is the symmetric coefficient vector, all of length $(2n+1-2k)$. As in [1], the unbiasedness constraint yields $\mathbf{1} \cdot \mathbf{a}_k = 1$. The MSE is $e_k = E[(\mathbf{n}_{(i),k} \cdot \mathbf{a}_k)^2] = \mathbf{a}_k^T \mathbf{H}_k \mathbf{a}_k$, with $\mathbf{H}_k$ the correlation matrix of $\mathbf{n}_{(i),k}$. Following [1], the optimal coefficients are given by $\mathbf{a}_k = (\mathbf{1}^T \mathbf{H}_k^{-1})/(\mathbf{1}^T \mathbf{H}_k^{-1} \mathbf{1})$, while the MMSE is $e_{k,min} = 1/(\mathbf{1}^T \mathbf{H}_k^{-1} \mathbf{1})$. The case $k = 0$ gives the results in [1].

Multiplicative noise

Now suppose that $\{x_i\}$ is a constant signal in multiplicative noise: $x_i = s n_i$, where n_i have a unity-mean, unity-variance distribution. We consider computation of the unbiased k-censored filter minimizing the normalized MSE $e_k = E[(y_i - s)^2 / s^2]$. The output of a k-censored filter with input $\{x_i\}$ is $y_i = s(\mathbf{n}_{(i),k} \cdot \mathbf{a}_k)$. The unbiasedness constraint now requires that $E[\mathbf{n}_{(i),k} \cdot \mathbf{a}_k] = 1$; since we do not imposing symmetry on the noise distribution, no simple constraint on the coefficients is obtained. The MSE is given by $e_k = \mathbf{a}_k^T \mathbf{H}_k \mathbf{a}_k - 1$. Denoting $\mathbf{m}_k = E[\mathbf{n}_{(i),k}]$, we form the Lagrangian $\text{L}(\mathbf{a}_k, \text{p}) = \mathbf{a}_k^T \mathbf{H}_k \mathbf{a}_k - 1 - p(\mathbf{m}_k \cdot \mathbf{a}_k - 1)$. Differentiating and equating to zero yields $2\mathbf{H}_k \mathbf{a}_k - p\mathbf{m}_k = 0$. Solving for $\mathbf{a}_k$ and p and substituting terms yields the optimal coefficients $\mathbf{a}_k = (\mathbf{m}_k^T \mathbf{H}_k^{-1})/(\mathbf{m}_k^T \mathbf{H}_k^{-1} \mathbf{m}_k)$. The MMSE is then $e_k = 1/(\mathbf{m}_k^T \mathbf{H}_k^{-1} \mathbf{m}_k) - 1$.

Computation and results

Most of the expense in computing the optimal $\mathbf{a}_k$ is incurred in calculating the matrices $\mathbf{H}_k$. However, the $\mathbf{H}_k$ are in fact submatrices of the correlation matrices for the uncensored case; these are available for many noise distributions and sample sizes.

We have computed MMSE k-censored filter coefficients and MSEs for six additive and one multiplicative (the negative-exponential) noise distributions, for odd values of n ranging from 3 to 25, and $1 \leq k \leq n$. Due to space limitations, the results will be given elsewhere. While the optimal censored filters are suboptimal relative to the optimal uncensored filters (of the same span), we have found that censored filters provide greater robustness against outlying observations, e.g., as measured by the MSE when the noise is heavy-tailed (Laplacian).

We have also computed MSEs for k-censored filters defined by adding k zero coefficients to optimal uncensored filters of length n. As expected, the resulting filters yield increased robustness; although these filters are suboptimal relative to k-censored filters of length $2(n+k)+1$, the differences in the MSE is usually small.

III. EDGE-PRESERVING PROPERTIES

We consider the edge-preserving merits of optimal OS filters with/without censoring. We make the simplifying assumption that the edge magnitude S is sufficiently large that the values on either side may be ordered separately and are independent, viz., the non-zero support of the symmetric density f is bounded to $(-b,b)$, $b \leq S/2$.

Suppose that a length-$(2n+1)$ filter is passed over a noisy step taking values 0 and S: at coordinate i the observations are $x_j = n_j$; $i-n \leq j \leq i-n+L-1$, and $x_j = n_j+S$; $i-n+L \leq j \leq i+n$. The bounded support of f implies that $\mathbf{x}_{(i)} = \mathbf{n}_1 + (\mathbf{n}_2 + S\mathbf{1}_2)$, where $\mathbf{n}_1 = (\mathbf{n}_{1(1)}, \ldots, \mathbf{n}_{1(L)} \mid 0)$ and $\mathbf{n}_2 = (0 \mid \mathbf{n}_{2(1)}, \ldots, \mathbf{n}_{1(2n+1-L)})$ are the lower and upper ordered noise samples with zero padding, and $\mathbf{1}_2 = (\mathbf{0} \mid \mathbf{1})$.

The output of a filter $\mathbf{a}$ is $y_i = \mathbf{a} \cdot (\mathbf{n}_1 + \mathbf{n}_2 + S\mathbf{1}_2)$. The MSE is $e_i(S) = E[y_i^2] = \mathbf{a}^T\mathbf{H}_{12}\mathbf{a} + S^2(\mathbf{a} \cdot \mathbf{1}_2)^2 + 2(\mathbf{a} \cdot \mathbf{m}_1)(\mathbf{a} \cdot \mathbf{m}_2) + 2S(\mathbf{a} \cdot \mathbf{m}_1 + \mathbf{a} \cdot \mathbf{m}_2)(\mathbf{a} \cdot \mathbf{1}_2)$, where $\mathbf{H}_{12} = E[\mathbf{n}_1{}^T\mathbf{n}_2]$, and $\mathbf{m}_j = E[\mathbf{n}_j]$, $j = 1,2$. For symmetry we minimize the average MSE $e_i = [e_i(S)+e_i(-S)]/2 = \mathbf{a}^T\mathbf{H}_{12}\mathbf{a} + 2(\mathbf{a} \cdot \mathbf{m}_1)(\mathbf{a} \cdot \mathbf{m}_2) + S^2(\mathbf{a} \cdot \mathbf{1}_2)^2$. Affixing the unbiasedness constraint $\mathbf{a} \cdot \mathbf{1} = 1$ (see [2]), we form the Lagrangian and proceed as before. In this case, however, the optimal $\mathbf{a}$ depends on the edge height S.

Results

We have computed the optimal censored and uncensored coefficients for several additive, symmetric noise distributions. The most interesting conclusion obtained is that the censored and uncensored filters are usually quite similar, except when extreme censoring was imposed. The reason for this is simply that the optimal uncensored filters tend to place greater weight on the centrally-ordered observations. Also, the MSEs obtained by zero-padding an optimal OS filter are usually similar to the optimal censored filter of the same overall length.

IV. CONCLUSIONS

We have shown coefficient censoring leads to increased robustness against outlying noise values and superior retention of edges. Indeed, censoring appears to be a significant attribute of OS filters contributing to their desireable smoothing properties. In particular, censored OS filters are of interest for image and signal filtering applications.

REFERENCES

[1] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., A generalization of median filtering using linear combinations of order statistics, *IEEE Trans. ASSP,* vol. 31, pp. 1342-1350, 1983.

[2] A.C. Bovik and L. Naaman, Least-squares signal estimation using order statistic filters, *Proc. Twentieth Ann. Conf. Info. Sci. Syst.,* March 1986.

[3] F. Palmieri and C.G. Boncelet, Optimal mse linear combination of order statistics for restoration of Markov processes, *Proc. Twentieth Ann. Conf. Info. Sci. Syst.,* March 1986.

[4] H.A. David, *Order Statistics.* New York, Wiley, 1981, p. 136.

[5] N.C. Gallagher and G.L. Wise, A theoretical analysis of the properties of median filters, *IEEE Trans. ASSP,* vol. 29, pp. 1136-1141, Dec. 1981.

[6] A. Restrepo and A.C. Bovik, Adaptive L-filters, *Proc. Twenty-Third Ann. Allerton Conf.,* Oct. 1985.

[7] A.C. Bovik, Streaking in median filtered images, *IEEE Trans. ASSP,* vol. 35, pp. 493-503, April 1987.

[8] H.G. Longbotham and A.C. Bovik, The equivalence of order statatistic and nonrecursive linear filters under monotonicity criteria, *Proc. Twenty-Fourth Ann. Allerton Conf.,* Oct. 1986.

AN ANALYSIS OF A MULTIPLE MODEL MORPHOLOGICAL FILTER

JISHENG SONG and EDWARD J. DELP
Purdue University
School of Electrical Engineering
Computer Vision and Image Processing Laboratory
West Lafayette, Indiana 47907

ABSTRACT

Rank order filters, including median and morphological filters, can be used to preserve details in an image and attenuate noise. In this paper, we analyze a morphological filter which reduces some of the blurring properties of previous used morphological filters due to limitations of the dimension and shape of the windows used. The properties of the filter are derived and experiments with images corrupted with impulsive noise are illustrated.

I. INTRODUCTION

Preserving geometric details, while suppressing noise in an image has been of interest in image processing research for a long time. Linear filtering algorithms, successfully used in 1-D signal processing and optimal for certain types of additive noise, will blur the fine details in an image. Rank order operators, including median and morphological filters, are nonlinear operators that can preserve fine details while suppressing noise[1, 2, 3, 4].

For the above two nonlinear operators, how well fine details of an image can be preserved depends upon the dimension and shape of the window used with the operators. To overcome the difficulty of preserving details, a multiple model approach has been utilized in both linear and nonlinear filtering schemes.

The basic idea of the approach is as follows: to the original image, we apply filters that extract information that represents different geometric features in the image. These different filters represent the models used to describe the image. A decision is then made at each pixel location relative to which model matches the image data in some sense. Figure 1 is a block diagram illustrating this approach.

The multiple model approach has been used with Kalman filtering[5]. Five models were used, one isotropic model, and four models for edges with different directions. Dravida[6] compared four filtering algorithms that included median filtering and the multiple model Kalman filter. Results of experiments indicated that the multiple model approach preserves more fine details than other filters. A multiple model approach, called max/median filtering, was proposed by Arce and Mcloughlin[7]. Four windows representing four lines were used and the final decision was made by choosing at each pixel location the maximum value of the four processed images.

In this paper, we analyze the multiple model approach based on morphological filters proposed by Stevenson and Arce[8, 9]. This algorithm is effective in suppressing noise, especially impulsive noise. Our analysis of the filter uses basic properties, such as antiextensive and extensive properties, of the morphological operators to derive new results. In [8, 9] the analysis was done from the point of view of transformations of the input signal, such as that of a constant and an oscillating signal.

This work was supported by the IBM Corporation, Federal Systems Division, Manassas, Virginia.

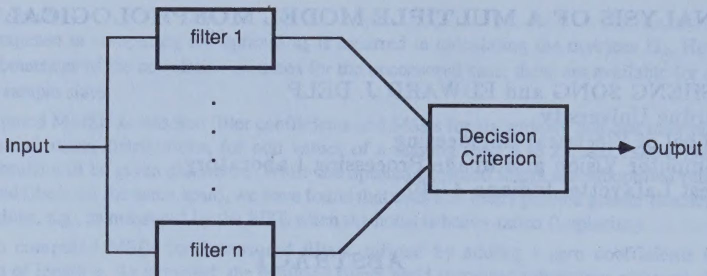


Figure 1. Block diagram of multiple model approach.

II. THE MULTIPLE MODEL MORPHOLOGICAL FILTER

Morphological filtering is a nonlinear transformation which can provide shape information about an image and can be used to preserve or remove specific geometric features in an image. The prior shape information, which is needed to perform the morphological transformation, is provided by another image called a structuring element[3]. The structuring element is basically an image with a simpler shape that is specified by the user. The various interactions of an image with a structuring element is the basis of morphological transformations. There are four basic morphological transformations known as erosion, dilation, opening and closing [3].

The morphological transformations of a binary image A by a structuring element B are defined as[3]:

$$\text{EROSION:} \quad A \ominus B = \left\{ c : \; c + b \in A \text{ for every } b \in B \right\}; \tag{1}$$

$$\text{DILATION:} \quad A \oplus B = \left\{ c : \; c = a + b \text{ for some } a \in A \text{ and } b \in B \right\}; \tag{2}$$

$$\text{OPENING:} \quad A \circ B = (A \ominus B) \oplus B; \tag{3}$$

$$\text{CLOSING:} \quad A \bullet B = (A \oplus B) \ominus B. \tag{4}$$

The morphological transformations of a gray scale image f by a gray scale structuring element g are defined as [3, 4]:

$$\text{EROSION:} \quad (f \oplus g)(x) = \min_{z \in G} \left\{ f(x+z) - g(z) \right\} \tag{5}$$

where G is the support area of the function g;

$$\text{DILATION:} \quad (f \ominus g)(x) = \max_{z \in G} \left\{ f(x-z) + g(z) \right\}; \tag{6}$$

$$\text{OPENING:} \quad f \circ g = (f \ominus g) \oplus g; \tag{7}$$

$$\text{CLOSING:} \quad f \bullet g = (f \oplus g) \ominus g. \tag{8}$$

Detailed descriptions of the above transformations can be found in [3, 10, 11].

Now we describe the multiple model morphological filter. The algorithm is composed of a cascade of two steps. The first step is defined by the following equation, f is the input image:

$$g = \max \left\{ f \circ L_0,\ f \circ L_{90},\ f \circ L_{45},\ f \circ L_{135} \right\}. \tag{9}$$

The final transformed image is denoted as y. The second step is defined as follows:

$$y = \min \left\{ g \bullet L_0,\ g \bullet L_{90},\ g \bullet L_{45},\ g \bullet L_{135} \right\}. \tag{10}$$

The shapes of the four structuring elements L_0, L_{90}, L_{45}, L_{135} are illustrated below:

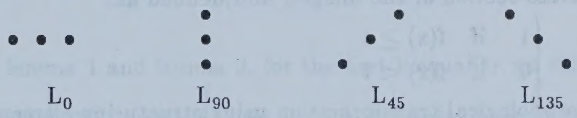

L_0
L_{90}
L_{45}
L_{135}

Figure 2 shows the block diagram of the multiple model morphological filter.

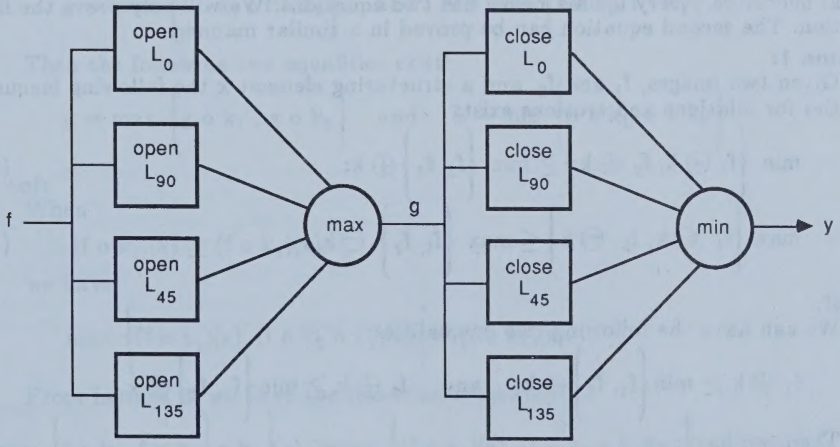

f
open L0
open L90
open L45
open L135
max
g
close L0
close L90
close L45
close L135
min
y

Figure 2. Block diagram of the multiple model morphological filter.

III PROPERTIES OF THE FILTER

In this section, we will derive some properties of the above filtering scheme and explain why the algorithm can remove noise without blurring details in the input image. First, as a background to following lemmas, we state several properties of basic morphological operations.

First are the antiextensive and the extensive properties[3, 10]. They are expressed by the following inequalities:

$$f \ominus L \leq f \circ L \leq f \leq f \bullet L \leq f \oplus L \tag{11}$$

Next is the idempotent property for both opening and closing[3, 10]:

$$f \circ L = (f \circ L) \circ L \quad \text{and} \quad f \bullet L = (f \bullet L) \bullet L. \tag{12}$$

The final property is that gray scale morphological transformations with binary structuring elements commute with thresholding:

$$X_t\,[\psi_L(f)] = \psi_L[X_t(f)]; \tag{13}$$

where X_t is the cross section of the image f and defined as:

$$X_t(f)(x) = \begin{cases} 1 & \text{if} \quad f(x) \geq t \\ 0 & \text{if} \quad f(x) < t \end{cases} \tag{14}$$

and ψ_L is the morphological transformation using structuring element L. The implication of (13) is that thresholding the transformed image is equivalent to transforming the thresholded original image. Proofs of above properties can be found in[3, 11].

The first property of the filter described in Figure 2 is that it is an idempotent operation, that is, the filtered image will be invariant if we apply the filter repeatedly. In order to prove this property, we must derive properties of the basic morphological operators. Every lemma below has two equations. We will only prove the first equation. The second equation can be proved in a similar manner.

Lemma 1:

Given two images, f_1 and f_2, and a structuring element k the following inequalities for dilations and erosions exist:

$$\min\left\{f_1 \oplus k,\, f_2 \oplus k\right\} \geq \min\left\{f_1,\, f_2\right\} \oplus k; \tag{15}$$

$$\max\left\{f_1 \ominus k,\, f_2 \ominus k\right\} \leq \max\left\{f_1,\, f_2\right\} \ominus k. \tag{16}$$

Proof:

We can have the following two inequalities:

$$f_1 \oplus k \geq \min\left\{f_1,\, f_2\right\} \oplus k; \quad \text{and} \quad f_2 \oplus k \geq \min\left\{f_1,\, f_2\right\} \oplus k.$$

Then, we have:

$$\min\left\{f_1 \oplus k,\, f_2 \oplus k\right\} \geq \min\left\{f_1,\, f_2\right\} \oplus k.$$

Lemma 2:

For two images, f_1 and f_2, and a structuring element k, the following equalities of dilation and erosion exist:

$$\max\left\{f_1 \oplus k,\, f_2 \oplus k\right\} = \max\left\{f_1,\, f_2\right\} \oplus k; \tag{17}$$

$$\min\left\{f_1 \ominus k,\ f_2 \ominus k\right\} = \min\left\{f_1,\ f_2\right\} \ominus k. \tag{18}$$

Proof:

These equalities can be easily proved by just changing the order of two operations, such as that of max with dilation.

Lemma 3:

For two images, f_1 and f_2, and a structuring element k, the following inequalities for opening and closing exist:

$$\max\left\{f_1 \circ k,\ f_2 \circ k\right\} \leq \max\left\{f_1,\ f_2\right\} \circ k \tag{19}$$

$$\min\left\{f_1 \bullet k,\ f_2 \bullet k\right\} \geq \min\left\{f_1,\ f_2\right\} \bullet k. \tag{20}$$

Proof:

Because of lemma 1 and lemma 2, for the first inequality we have

$$\left[\max\left\{f_1,\ f_2\right\} \ominus k\right] \oplus k \geq \max\left\{f_1 \ominus k,\ f_2 \ominus k\right\} \oplus k = \max\left\{f_1 \circ k,\ f_2 \circ k\right\}.$$

Lemma 4:

Given a image f and two structuring elements, k_1 and k_2,
let

$$g = \max\left\{f \circ k_1,\ f \circ k_2\right\}, \quad \text{and} \quad h = \min\left\{f \bullet k_1,\ f \bullet k_2\right\}.$$

Then the following two equalities exist:

$$g = \max\left\{g \circ k_1,\ g \circ k_2\right\} \quad \text{and} \quad h = \min\left\{h \bullet k_1,\ h \bullet k_2\right\}. \tag{21}$$

Proof:

When

$$(f \circ k_1)(x) \geq (f \circ k_2)(x),$$

we have

$$\max\left\{(f \circ k_1)(x),\ (f \circ k_2 \circ k_1)(x)\right\} = (f \circ k_1)(x).$$

From Lemma 3, we have the following inequality:

$$\left[\max\left\{f \circ k_1,\ f \circ k_2\right\} \circ k_1\right](x) \geq \max\left\{(f \circ k_1)(x),\ (f \circ k_2 \circ k_1)(x)\right\} = (f \circ k_1)(x).$$

From the antiextensive property of opening, we have the following inequality:

$$\left(\max\left\{f \circ k_1,\ f \circ k_2\right\} \circ k_1\right)(x) \leq (f \circ k_1)(x)$$

Then, from above two inequalities we have

$$\left(\max\left\{f \circ k_1,\ f \circ k_2\right\} \circ k_1\right)(x) = (f \circ k_1)(x).$$

So the following equality exists in this case:

$$\max \left\{ (g \circ k_1)(x),\, (g \circ k_2)(x) \right\} = (f \circ k_1)(x).$$

Similarly, when

$$(f \circ k_2)(x) \geq (f \circ k_1)(x),$$

we have

$$\max \left\{ (g \circ k_1)(x),\, (g \circ k_2)(x) \right\} = (f \circ k_2)(x).$$

So

$$\max \left\{ (g \circ k_1)(x),\, (g \circ k_2)(x) \right\} = \max \left\{ (f \circ k_1)(x),\, (f \circ k_2)(x) \right\} = g(x).$$

Now we can state the idempotent property of the filter.

Theorem I:

Denote

$$f_{k^*} = \max \left\{ f \circ k_1,\, f \circ k_2,\, f \circ k_3,\, f \circ k_4 \right\},\quad f^{k^*} = \min \left\{ f \bullet k_1,\, f \bullet k_2,\, f \bullet k_3,\, f \bullet k_4 \right\},$$

and let

$$y = (f_{k^*})^{k^*}.$$

Then

$$y = (y_{k^*})^{k^*}. \tag{22}$$

Proof:

$$y = y^{k^*} \geq (y_{k^*})^{k^*} = \left[((f_{k^*})^{k^*})_{k^*} \right]^{k^*} \geq \left[(f_{k^*})_{k^*} \right]^{k^*} = (f_{k_*})^{k^*} = y.$$

The filter is idempotent.

The threshold decomposition and stacking properties of rank order filters provides not only a new and efficient VLSI implementations, but also a new tool to analyze these nonlinear filters[12]. A necessary and sufficient condition for a binary function to satisfy the stacking property is that it can be expressed as a Boolean expression which contains no complements of the input variables. Such binary functions are called positive Boolean functions[12].

First, we maintain that the multiple model morphological filter possesses the threshold decomposition and stacking property.

Theorem II:

The multiple model morphological filter possesses the thresholding decomposition and stacking property.

Proof:

We know that opening and closing commute with thresholding as stated at the beginning of this section[3, 11]. Maximum and minimum operations also commute with thresholding [12]. So the cascade and parallel combination of them possesses the property of thresholding decomposition.

The filtering algorithm operating on a cross section of an image f can be defined as:

$$X_t(g) = (X_t(f) \circ L_0) \bigcup (X_t(f) \circ L_{90}) \bigcup (X_t(f) \circ L_{45}) \bigcup (X_t(f) \circ L_{135}), \tag{23}$$

and

$$X_t(y) = (X_t(g) \bullet L_0) \cap (X_t(g) \bullet L_{90}) \cap (X_t(g) \bullet L_{45}) \cap (X_t(g) \bullet L_{135}) \,. \qquad (24)$$

It is clear that above equations are positive Boolean functions.

The impulsive noise in an image usually appears as discrete isolated pixel variations. Values of noise pixels often markably differ from their neighbors. So after thresholding, the noise will appear to be isolated pixels surrounded by neighbors having opposite values. This leads to the following property for binary images.

Theory III:

The value of a pixel, zero or one, that has at least N pixels of the same value on both sides of itself along one of the four directions will be invariant after applying the filter.

Proof:

This proof is demonstrated with 3x3 structuring elements. It can be extended to the general case of structuring elements with (2N+1)x(2N+1) windows. A 5x5 image region is shown as follows:

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

where, the pixel x_{ij} takes on a value of either 1 or 0.

Denote the pixels after the first step of the filter as z_{ij}. For example, the expression for z_{33} is defined as:

$$\begin{aligned} z_{33} = &\left\{ \Big(\bigcap_{i=1}^{3} x_{3i}\Big) \cup \Big(\bigcap_{i=2}^{4} x_{3i}\Big) \cup \Big(\bigcap_{i=3}^{5} x_{3i}\Big) \right\} \cup \left\{ \Big(\bigcap_{i=1}^{3} x_{i3}\Big) \cup \Big(\bigcap_{i=2}^{4} x_{i3}\Big) \cup \Big(\bigcap_{i=3}^{5} x_{i3}\Big) \right\} \cup \\ &\left\{ \Big(\bigcap_{i=1}^{3} x_{i,(6-i)}\Big) \cup \Big(\bigcap_{i=2}^{4} x_{i,(6-i)}\Big) \cup \Big(\bigcap_{i=3}^{5} x_{i,(6-i)}\Big) \right\} \cup \left\{ \Big(\bigcap_{i=1}^{3} x_{ii}\Big) \cup \Big(\bigcap_{i=2}^{4} x_{ii}\Big) \cup \Big(\bigcap_{i=3}^{5} x_{ii}\Big) \right\} \qquad (25) \end{aligned}$$

Note that for z_{33} to have the value 1, at least one of 12 logical AND terms must be nonzero. Each of the logical terms consists of a line segment of 3 pixels in length that passes through x_{33}. Hence, the geometrical interpretation of (25) is that z_{33} is 1 if there is a bright 3 pixel line passing through x_{33}. If x_{33} is an isolated impulsive noise pixel, z_{33} will have the value 0. Therefore positive impulsive noise is suppressed and bright geometric details are preserved after this step. We can also observe that along edges of bright areas no zero are ever created. Therefore, there is no smearing or blurring after this step.

Now we continue to study the second processing step of the filter. The 5x5 image region is illustrated as follows:

z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
z_{21}	z_{22}	z_{23}	z_{24}	z_{25}
z_{31}	z_{32}	z_{33}	z_{34}	z_{35}
z_{41}	z_{42}	z_{43}	z_{44}	z_{45}
z_{51}	z_{52}	z_{53}	z_{54}	z_{55}

Denote the pixels after the second step of the filter as y_{ij}. For example, the expression for pixel y_{33} is defined as follows:

$$y_{33} = \left\{ \Big(\bigcup_{i=1}^{3} z_{3i}\Big) \cap \Big(\bigcup_{i=2}^{4} z_{3i}\Big) \cap \Big(\bigcup_{i=3}^{5} z_{3i}\Big) \right\} \cap \left\{ \Big(\bigcup_{i=1}^{3} z_{i3}\Big) \cap \Big(\bigcup_{i=2}^{4} z_{i3}\Big) \cap \Big(\bigcup_{i=3}^{5} z_{i3}\Big) \right\} \cap$$
$$\left\{ \Big(\bigcup_{i=1}^{3} z_{i,(6-i)}\Big) \cap \Big(\bigcup_{i=2}^{4} z_{i,(6-i)}\Big) \cap \Big(\bigcup_{i=3}^{5} z_{i,(6-i)}\Big) \right\} \cap \left\{ \Big(\bigcup_{i=1}^{3} z_{ii}\Big) \cap \Big(\bigcup_{i=2}^{4} z_{ii}\Big) \cap \Big(\bigcup_{i=3}^{5} z_{ii}\Big) \right\} \qquad (26)$$

Note that for y_{33} to have the value 0, at least one of 12 logical OR terms must be zero. Also each of the logical terms consists of a line segment of 3 pixels in length that passes through z_{33}. Hence, the geometrical interpretation of (26) is that y_{33} is 0 if there is a dark 3 pixel line passing through z_{33}. If z_{33} is an isolated impulsive noise pixel, y_{33} will have the value 1. Therefore negative impulsive noise is suppressed and dark geometric details are preserved after this step. We can also observe that along edges of dark areas no one is ever created. Therefore, there is no smearing or blurring after this step.

The above property of this filter is illustrated by Figure 3.

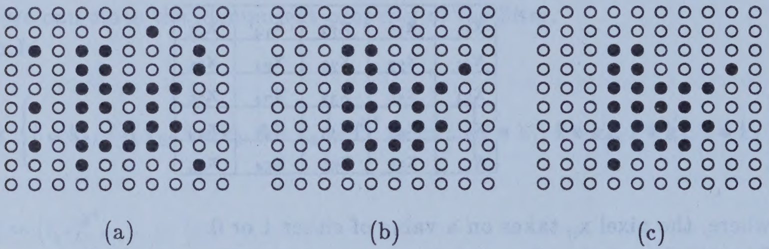


Figure 3. (a) image corrupted with impulsive noise; (b) the result after the first computation; (c) the final result.

We applied the filter to an image corrupted with impulsive noise. Median filtering, the max/median filter, the opening-closing morphological filter and the multiple model morphological filter are used to process the noisy image. The results are shown in Figure 4.

IV CONCLUSION

In this paper, a morphological filter based on a multiple model has been analyzed. Four line models are represented by four structuring elements which are used to perform the morphological transformations. We proved that the filter has the idempotent property. Hence no iterations are necessary so that the amount of computation required by the filter is relatively small. We also showed that the filter possesses the threshold decomposition property. By using the threshold decomposition property, we carried out a deterministic analysis of the binary version of the filter to demonstrate the effects of the filter on image structure.

References

1. T. S. Huang and G. J. Yang, *Median filters and their applications to image processing,* School of Electrical Engineering, Purdue University, West Lafayette, Indiana; Technical Report EE 80-1, Jan. 1980.
2. N. C. Gallagher and G. L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-29 No. 6, pp. 1136-1141, Dec. 1981.
3. J. Serra, *Image Analysis and Mathematical Morphology,* Academic Press, 1982.
4. Stanley R. Sternberg, "Biomedical image processing," *IEEE Computer,* vol. 16, pp. 22-34, Jan. 1983.

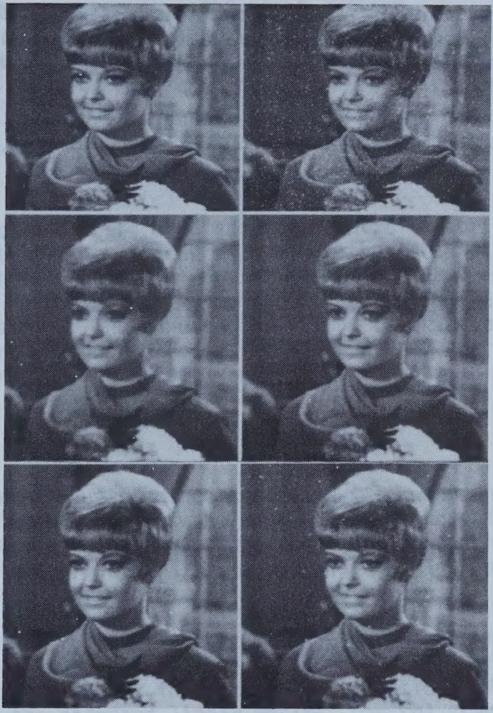


Figure 4. *Upper left*: the original image; *Upper right*: the image corrupted with impulsive noise; *Center left*: the opening-closing morphological filter with a 3x3 square structuring element; *Center right*: the median filter with a 3x3 window; *Lower left*: the multiple model morphological filter; *Lower right*: the max/median filter.

5. R. Mediavilla, H. Kaufman, M. Tekalp, and J. W. Woods, "Multiple models for nonstationary image estimation," *Proc. 6th IFAC Symposium on Identification and System Parameter Estimation*, pp. 1467-1471, June 1982.

6. S. Dravida, J. W. Woods, and W. C. Shen, "A comparison of image filtering algorithms," *Proc. 1984 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 23.3.1-23.3.4, March 1985.

7. G. R. Arce and M. P. Mcloughlin, "Theoretical analysis of the max/median filter," *IEEE Trans. on Acoustics, Speech, and Signal Processing.*, vol. ASSP-35, No. 1, pp. 60-69, January 1987.

8. R. L. Stevenson and G. R. Arce, "Theoretical analysis of morphological filters," *Proc. 24th Annual Allerton Conference on Communication, Control, and Computing*, pp. 353-362, Oct. 1986.

9. R. L. Stevenson and G. R. Arce, "Theoretical analysis of morphological filters," *IEEE Trans. on Circuit and Systems*, to appear.

10. R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, No. 4, pp. 532-550, July 1987.

11. P. A. Maragos and R. W. Schafer, "A unification of linear, median, order statistics, and morphological filters under mathematical morphology," *Proc. 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1985.
12. P. D. Wendt, E. J. Coyle, and N. C. Gallagher, "Stack filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing.*, vol. ASSP-34, No. 4, pp. 898-911, August 1986.

DECENTRALIZED ADAPTIVE CONTROL OF LARGE SCALE SYSTEMS

H. YOUSEF and M. SIMAAN
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261

ABSTRACT

A decentralized model reference adaptive control algorithm for MIMO interconnected systems is developed. The algorithm does not require identification of the system parameters, or satisfaction of perfect model following conditions. Furthermore, it imposes no additional restrictions on the system output matrix. The boundedness of the tracking error and the controller parameters are guaranteed using the Lyapunov approach. A numerical example illustrating the effectiveness of the proposed algorithm is presented.

I. INTRODUCTION

The design of a high-performance control system when the plant dynamics are poorly known requires the application of adaptive control techniques. Among these, model reference adaptive control has been extensively used due to the ease of implementation and high speed of adaptation [1].

The adaptive model-following control (AMFC) is one of the most effective models which has been utilized to overcome the difficulty of expressing real design objectives in terms of a quadratic performance index - to be able to apply the standard linear optimal control theory - for a plant whose parameters are unknown or varying during the operation [2]. Landau and Courtiol [3] have developed an adaptive control design algorithm which requires satisfaction of perfect model-following conditions (PMFC) first established by Erzberger [4]. Sobel et al. [5] have developed an adaptive control algorithm which ensures asymptotic stability of the error between plant and model outputs without satisfaction of the PMFC. This algorithm, however, requires the plant output matrix to satisfy an additional condition in terms of the solution of a Lyapunov matrix equation. The problem of designing adaptive control for decentralized systems has received the attention of many researchers during the last few years [6-8].

The purpose of this paper is to discuss a design procedure for a decentralized adaptive control of a large-scale interconnected system whose parameters are varying during the operation. This procedure uses the implicit model reference adaptive approach in such a way that the outputs of the plant are required to track asymptotically the outputs of the model. Furthermore, this design does not impose the additional restriction on the plant output matrix as required in [5].

II. PROBLEM FORMULATION

Consider a large-scale system which is described as an interconnection of N subsystems

$$\dot{x}_{pi} = A_{pi}x_{pi} + B_{pi}u_{pi} + \sum_{\substack{j=1\\ \neq i}}^{N} A_{ij}x_{pj} \qquad i=1,2,\ldots,N \tag{1}$$

$$y_{pi} = C_{pi}x_{pi} \tag{2}$$

where for the ith subsystem, $x_{pi}\epsilon R^{ni}$ is the state vector, $u_{pi}\epsilon R^{ri}$ is the input control vector, and $y_{pi}\epsilon R^{qi}$ is the measurable output vector. The matrices A_{pi}, B_{pi}, A_{ij} and C_{pi} are constant matrices of appropriate dimensions. The values, or some values, of any (or all) of the matrices A_{pi}, B_{pi} and A_{ij} are varying during the operation in an unknown fashion. However, the admissible parameters of these matrices are assumed to be bounded. Furthermore, it is also assumed that all pairs (A_{pi}, B_{pi}) in the bounded range are controllable and output stabilizable and all the pairs (A_{pi}, C_{pi}) are observable. The composite system can be described as

$$\dot{x}_p = A_p x_p + B_p u_p + Hx_p \tag{3}$$

$$y_p = C_p x_p \tag{4}$$

where $x_p=[x_{p1}^T,\ldots,x_{pN}^T]^T$, $u_p=[u_{p1}^T,\ldots,u_{pN}^T]^T$ and $y_p=[y_{p1}^T,\ldots,y_{pN}^T]^T$ are the overall state vector, control vector and output vector respectively. The matrices A_p, B_p and C_p are block diagonal and the matrix H has zeroes on the diagonal, A_{ij} on the upper diagonal and A_{ji} on the lower diagonal.

The problem is to design local adaptive controllers using only local information such that the subsystem output vector tracks the output of the corresponding stable local reference model.

$$\dot{x}_{mi} = A_{mi}x_{mi} + B_{mi}u_{mi} \tag{5}$$

$$i=1,2,\ldots,N$$

$$y_{mi} = C_{mi}x_{mi} \tag{6}$$

where $x_{mi}\epsilon R^{mi}$, $u_{mi}\epsilon R^{ri}$ and $y_{mi}\epsilon^{qi}$.

The concept of command generator tracker (CGT) [9] is utilized to facilitate the development of the adaptive control algorithm. When perfect tracking occurs, the ideal state and control vector for the ith subsystem, x^*_{pi} and u^*_{pi}, satisfy the output equation and the dynamical equation of the decoupled subsystem. Expressed mathematically, this means that:

$$C_{pi}x^*_{pi} = C_{mi}x_{mi} \Rightarrow y^*_{pi} = y_{mi} \tag{7}$$

and

$$\dot{x}^*_{pi} = A_{pi}x^*_{pi} + B_{pi}u^*_{pi} \tag{8}$$

Moreover, x^*_{pi} and u^*_{pi} are assumed to be linear function of the submodel state vector x_{mi} and the submodel input vector u_{mi} [9].

$$x^*_{pi} = S_{1i}x_{mi} + S_{2i}u_{mi} \tag{9}$$

$$u^*_{pi} = S_{3i}x_{mi} + S_{4i}u_{mi} \tag{10}$$

where S_{ki}, k=1,2,3, and 4 are the feedforward matrices for the ith subsystem. The evaluation of these matrices depends on ri and qi, the number of subsystem input and output respectively. A procedure for finding these matrices when $r_i \geq q_i$ is discussed in [9]. The case when $r_i < q_i$ does not have a solution.

The error vector $e_i(t)$ is defined as:

$$e_i = x^*_{pi} - x_{pi} \tag{11}$$

when $\lim_{t\to\infty} e_i = 0$, we get $x^*_{pi} = x_{pi}$ which gives $y^*_{pi} = y_{pi}$. But by definition, $y^*_{pi} = y_{mi}$, therefore $y_{mi} = y_{pi}$ which is the required objective.

III. CONTROLLER STRUCTURE

The controller structure for the ith subsystem is given by [5]

$$u_{pi} = K_i r_i \tag{12}$$

where $r_i^T = [(C_{pi} e_i)^T, x_{mi}^T, u_{mi}^T]$ (13)

and $K_i = [K_{ei}, K_{xi}, K_{ui}]$ (14)

The gain K_i is chosen to be a sum of proportional gain K_{pi} and an integral gain K_{Ii} each of which is adapted as follows [5]:

$$K_i = K_{pi} + K_{Ii} \tag{15}$$

$$K_{pi} = g_i r_i^T F_i \tag{16}$$

$$\dot{K}_{Ii} = g_i r_i^T G_i \quad , \ K_{Ii}(0) = K_{Ii0} \tag{17}$$

and $g_i = D_i e_{i0}$ (18)

where the weighting matrices F_i and G_i are positive definite matrices and the processing matrix D_i is chosen in such a way to guarantee the stability of the error equation. The error vector e_{i0} is defined as:

$$e_{i0} = x^*_{pi0} - x_{pi0}$$

where x^*_{pi0} and x_{pi0} are nominal values of the ideal plant state vector and the actual plant state vector respectively.

IV. ERROR EQUATION AND STABILITY ANALYSIS

Differentiating (11) and using (1) and (8), we get

$$\dot{e}_i = A_{pi} e_i + B_{pi}(u^*_{pi} - u_{pi}) + \sum_{\substack{j=1 \\ \neq i}}^{N} A_{ij}(e_j - x^*_{pj}) \tag{19}$$

Now, substituting (11), (12), (15), (16) and (18) into (19) we get

$$\dot{e}_i = A_{pi}e_i + B_{pi}(S_{3i}x_{mi} + S_{4i}u_{mi} - K_{Ii}r_i - D_i e_{i0} r_i^T F_i r_i) + \sum_{\substack{j=1\\ \neq i}}^{N} A_{ij}(e_j - x^*_{pj}) \tag{20}$$

and for the decoupled subsystems

$$\dot{e}_i = A_{pi}e_i + B_{pi}(S_{3i}x_{mi} + S_{4i}u_{mi} - K_{Ii}r_i - D_i e_{i0} r_i^T F_i r_i) \tag{21}$$

The following theorem is concerned with the boundedness of $e_i(t)$ and $K_{Ii}(t)$ for the decoupled subsystem.

Theorem 1: For $i=1,2,\ldots,N$ let $\varepsilon_i = e_{i0} - e_i$, B_{pi} is a full rank matrix, $\lambda_{mi} = \min\lambda(Q_i)$ and $\lambda_{Mi} = \max\lambda(D_i D_i^T)$ where

$$D_i = B_{pi}^T P_i \quad , \tag{22}$$

P_i is the positive definite solution of Lyapunov matrix equation

$$P_i(A_{pi} - B_{pi}\tilde{K}_{Iei}C_{pi}) + (A_{pi} - B_{pi}\tilde{K}_{Iei}C_{pi})^T P_i = -Q_i \tag{23}$$

and Q_i is a positive definite matrix. Then, the solution $e_i(t)$ and $K_{Ii}(t)$ of (21) and (17) is uniformly ultimately bounded. Moreover, the tracking error $e_i(t)$ converges to the residual set $\mathrm{Resd} = \{e_i : \|e_i\| < \gamma\}$ where:

$$\gamma < \left(\frac{2d_0}{\lambda_{mi}}\right)^{\frac{1}{2}}$$

and

$$d_0 = 2\,\frac{\lambda_{Mi}}{\lambda_{mi}}\,\|\varepsilon_i\|^2\,\|K_{Ii} - \tilde{K}_{Ii}\|^2 \sup_t\left(\frac{\|r_i\|^2}{\|e_i\|^2}\right)$$

The proof of theorem 1 is presented in Appendix A. Using (22) and (23), the following theorem considers the stability property of (20) and (17).

Theorem 2: Let $\delta_{ij} = \|P_i A_{ij}\|$ where P_i is given by (23). If there exists an N vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T > 0$ such that the NxN matrix W whose elements are given by

$$\omega_{ij} = \begin{cases} \alpha_i\lambda_{mi} & i=j \\ -(\alpha_i\delta_{ij} + \alpha_j\delta_{ji}) & i\neq j \end{cases} \tag{24}$$

is positive definite, then the solution $e_i(t)$ and $K_{Ii}(t)$ of (20) and (17) is uniformly ultimately bounded and the overall tracking error $e(t)$ converges to the residual set

$$R_{esi} = \{e : \|e\| < \xi\}$$

where $\xi = \left(\frac{2d_0}{\lambda_m}\right)^{\frac{1}{2}}$

and $$d_0 = \frac{4}{\lambda_m} [\lambda_{M1} ||\alpha\varepsilon||^2 ||K_I - \tilde{K}_I||^2 \frac{||r||^2}{||e||^2} + \lambda_{M2}^2 ||\alpha H||^2 ||x_p^*||^2]$$

where $\lambda_m = \min \lambda(W)$
$\lambda_{M1} = \max \lambda(DD^T)$
$\lambda_{M2} = \max \lambda(P)$ and D and P are block diagonal matrices. The proof of theorem 2 is given in Appendix B.

V. REMARKS

Some remarks about the algorithm are mentioned below.

1. The plant parameters are assumed to be varying during the operation (unknown variation for the designer) and the range of variations is assumed to be bounded (known bounded set).

2. The design of the matrix D_i (22) requires a prior knowledge of the plant matrices A_{pi} and B_{pi}. To overcome this problem, the matrix D_i is determined by solving the standard LQR problem using a nominal value of the plant matrices belonging to the bounded set of parameter variations. The robustness of this matrix in the sense of satisfying (22) and (23) for deviations in A_{pi} and B_{pi} must then be checked.

3. The nominal value of the ideal plant state vector x_{pi0}^* and the actual plant state vector x_{pi0} should be evaluated in such a way to yield $\min(\varepsilon_i)$ and hence to decrease the size of the residual set.

4. The positive definiteness of the matrix W cannot be easily checked since it requires prior knowledge of the plant matrices A_{pi}, B_{pi} and A_{ij}. Therefore, the results of theorem 2 should be interpreted qualitatively [10].

VI. EXAMPLE

To illustrate the effectiveness of the proposed algorithm we consider a two interconnected second order subsystems,

S1:

$$\dot{x}_{p1} = \begin{bmatrix} \alpha_{11} & 1.0 \\ 2.0 & \alpha_{22} \end{bmatrix} x_{p1} + \begin{bmatrix} 1.0 \\ 0.6 \end{bmatrix} u_{p1} + \begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.2 \end{bmatrix} x_{p2}$$

$$y_{p1} = [0.8 \quad 1.2] x_{p1}$$

S2:

$$\dot{x}_{p2} = \begin{bmatrix} \alpha_{33} & 1.5 \\ 3.0 & \alpha_{44} \end{bmatrix} x_{p2} + \begin{bmatrix} 1.0 \\ 0.25 \end{bmatrix} u_{p2} + \begin{bmatrix} 0.1 & 0.2 \\ 0.8 & 0.5 \end{bmatrix} x_{p1}$$

$$y_{p2} = [1.0 \quad 0.2] x_{p2}$$

where

$-3.0 \leq \alpha_{11} \leq -1.8$, $-6.0 \leq \alpha_{22} \leq -4.0$, $-10.0 \leq \alpha_{33} \leq -8.0$ and $-4.0 \leq \alpha_{44} \leq -2.5$

The overall output $[y_{p1} \quad y_{p2}]$ is required to track asymptotically the overall output $[y_{m1} \quad y_{m2}]$ of the following reference model.

$$\dot{x}_M = \begin{bmatrix} -5.0 & 5.0 & 0.0 & 0.0 \\ -1.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -2.0 & -1.0 \\ 0.0 & 0.0 & 5.0 & -0.5 \end{bmatrix} x_M + \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 0.0 \end{bmatrix} u_M$$

$$Y_M = \begin{bmatrix} 0.8 & 1.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.2 \end{bmatrix} X_M$$

where

$$u_M = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$$

A stepwise parameter variation of A_p is considered for the composite system between two conditions described by
Condition #1

$$A_p = \begin{bmatrix} -3.0 & 1.0 & 0.0 & 0.0 \\ 2.0 & -4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -8.0 & 1.5 \\ 0.0 & 0.0 & 3.0 & -2.5 \end{bmatrix}$$

and Condition #2

$$A_p = \begin{bmatrix} -1.8 & 1.0 & 0.0 & 0.0 \\ 2.0 & -6.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -10.0 & 1.5 \\ 0.0 & 0.0 & 3.0 & -4.0 \end{bmatrix}.$$

The simulation results are shown in Fig. (1-5) for a 30-sec interval where the stepwise variation of A_p occurs at time t=15 sec. It is obvious from Fig. (1-3) that the tracking error e_{01} and e_{02} are bounded. Fig. (4-5) shows the boundedness of the controller parameters.

VII. CONCLUSION

In this paper we developed a decentralized model reference adaptive control algorithm for an interconnected system. Our algorithm does not require satisfaction of the PMF conditions or identification of the plant parameters. By introducing a linear transformation on the error vector $e_{i0}(t)$, it also eliminates the constraint imposed on the plant output matrix. The algorithm guarantees boundedness of the tracking error and the controller paramters. The size of the residual set depends on the bounds of the interconnection and ε_i. A numerical example is presented which shows the effectiveness of the proposed algorithm.

APPENDIX A

Proof of theorem 1:

Consider the positive definite function

$$v_i(t) = e_i^T P_i e_i + tr[(K_{Ii} - \tilde{K}_{Ii}) G_i^{-1} (K_{Ii} - \tilde{K}_{Ii})^T] \quad \text{(A1)}$$

The time derivative of $v_i(t)$ along the solution of (17) and (21) is

$$\dot{v}_i(t) = -e_i^T Q_i e_i - 2e_i^T (D_i^T D_i) e_{i0} r_i^T F_i r_i + 2\varepsilon_i^T D_i^T (K_{Ii} - \tilde{K}_{Ii}) r_i \quad \text{(A2)}$$

where in (A2) we make use of (22) and (23). Taking the norm of (A2), knowing that B_{pi} is a full rank yields,

$$\dot{v}_i(t) \leq - \lambda_{mi} ||e_i||^2 + 2||\varepsilon_i|| \, [||K_{Ii} - \tilde{K}_{Ii}||] \, ||r_i|| \, ||D_i^T|| \quad \text{(A3)}$$

which gives after completing the square,

$$\dot{v}_i(t) \leq - \frac{\lambda_{mi}}{2} ||e_i||^2 + 2 \frac{\lambda_{Mi}}{\lambda_{mi}^2} \frac{||\varepsilon_i r_i||^2}{||e_i||^2} ||K_{Ii} - \tilde{K}_{Ii}||^2 \quad \text{(A4)}$$

where $||D_i^T|| = \lambda_{Mi}^{\frac{1}{2}}$

In view of (A1) and (A4), the solution of $e_i(t)$ and $K_{Ii}(t)$ of (21) and (17) is uninformly ultimately bounded [10] and the residual set is given by [6].

$$R_{esd} = \{e_i : ||e_i|| < \gamma\}$$

where $\gamma < (\frac{2d_0}{\lambda_{mi}})^{\frac{1}{2}}$

and $d_0 = \frac{2\lambda_{Mi}}{\lambda_{mi}} ||\varepsilon_i||^2 \, ||K_{Ii} - \tilde{K}_{Ii}|| \sup_t \frac{||r_i||^2}{||e_i||^2}$

APPENDIX B

Proof of theorem 2:

Consider the positive definite function

$$V(t) = \sum_{i=1}^{N} \alpha_i v_i(t) \quad \text{(B1)}$$

where $v_i(t)$ is given in (A1). The time derivative of (B1) along the solution (17) and (20) is

$$\dot{V}(t) = \sum_{i=1}^{N} \alpha_i[-e_i^T Q_i e_i - 2e_i^T(D_i^T D_i)e_{10} r_i^T F_i r_i + 2\varepsilon_i^T D_i^T (K_{Ii} - \tilde{K}_{Ii}) r_i$$

$$+ 2e_i^T P_i \sum_{\substack{j=1 \\ \neq i}}^{N} A_{ij}(e_j - x_{pj}^*)] \tag{B2}$$

Taking the norm of (B2) and using (24) we get,

$$\dot{V}(t) \leq -\frac{\lambda_m}{2} ||e||^2 + r \frac{\lambda_{M1}}{\lambda_m} ||\varepsilon\alpha||^2 \frac{||K_I - \tilde{K}_I||^2}{||e||^2} ||r||^2$$

$$+ 4 \frac{\lambda_{M2}^2}{\lambda_m} ||\alpha H||^2 ||x_p^*||^2 \tag{B3}$$

where $\varepsilon=[\varepsilon_1,\dots,\varepsilon_N]$. Since W is positive definite, the solution of (20) and (17) is uniformly ultimately bounded [10] and the residual set is given by

$$R_{esi} = \{e: ||e|| < \xi\}$$

where $\xi = (\frac{2d_0}{\lambda_m})^{\frac{1}{2}}$

and $d_0 = \frac{4}{\lambda_m} [\lambda_{M1} ||\alpha\varepsilon||^2 ||K_I - \tilde{K}_I||^2 \frac{||r||^2}{||e||^2} + \lambda_{M2}^2 ||\alpha H||^2 ||x_p^*||^2]$

REFERENCES

[1] Landau, I.D., (1979), Adaptive Control: The Model Reference Approach, Marcel Dekker, Inc., NY.

[2] Landau, I.D., (1974), "A Survey of Model Reference Adaptive Techniques -Theory and Applications," Automatica, Vol. 10, pp. 353-379.

[3] Landau, I.D. and Courtiol, B., (1974), "Design of Multivariable Adaptive Model-Following Control Systems," Automatica, Vol. 10, pp. 483-494.

[4] Erzberger, H., (1968), "On the Use of Algebraic Methods in the Analysis and Design of Model-Following Control Systems," NASA Ames Research Center, Report #TND-4663.

[5] Sobel, K., Kaufman, H. and Mabius, L., (1982), "Implicit Adaptive Control for a Class of MIMO Systems," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-18, pp. 576-589.

[6] Siljak, D.D. and Gavel, D.T., (1985), "Adaptive Control of Decentralized Systems: Known Subsystems, Unknown Interconnections," IEEE Conference on Decision and Control, Ft. Lauderdale, FL, pp. 1858-1861.

[7] Al-Abbass, F. and Ozguner, U., (1985), "Decentralized Model Reference Adaptive System Using a Variable Structure Control," IEEE Conference on Decision and Control, Ft. Lauderdale, FL, pp. 1473-1478.

[8] Ioannau, P., (1986), "Decentralized Control of Interconnected Systems," IEEE Transactions on Automatic Control, Vol. AC-31, pp. 291-298.

[9] O'Brien, M.J. and Broussard, J.R., (1980), "Feedforward Control to Track the Output of a Forced Model," IEEE Transactions on Automatic Control, Vol. AC-25, pp. 851-853.

[10] Michel, A.N. and Miller, Qualitative Analysis of Large Scale Dynamical Systems, NY, Academic Press, 1977.

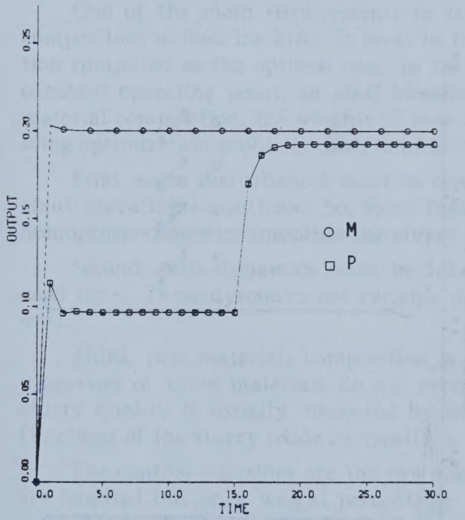


Figure 1. Outputs of subsystem and submodel # 1

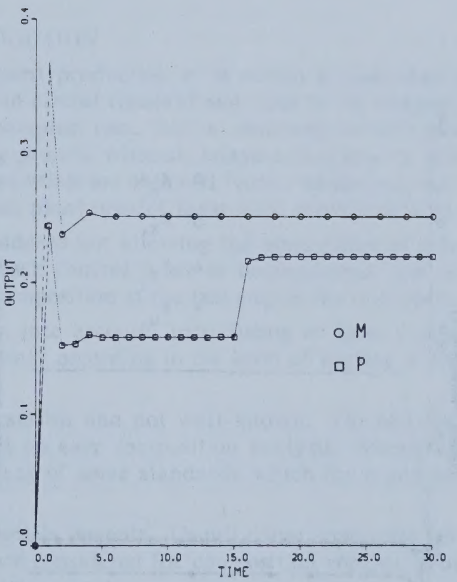


Figure 2. Outputs of subsystem and submodel # 2

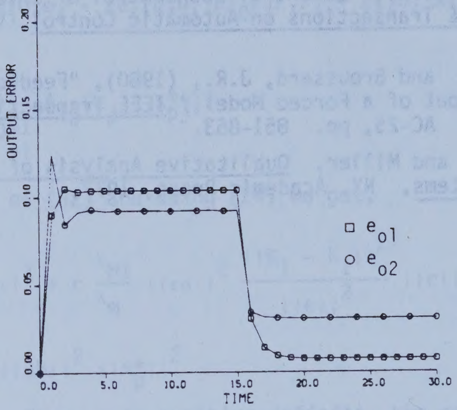


Figure 3. Output errors for subsystems 1 & 2

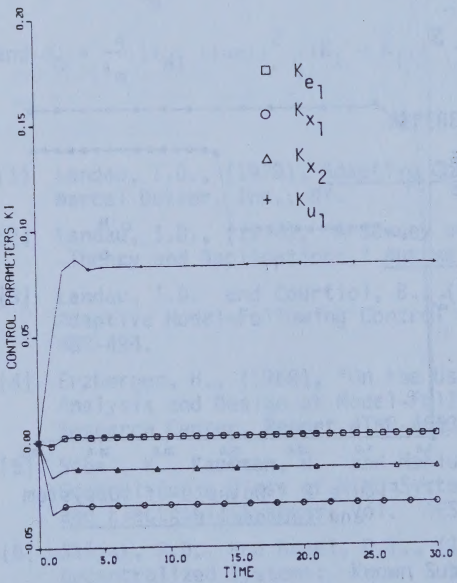


Figure 4. Control parameters for subsystem # 1

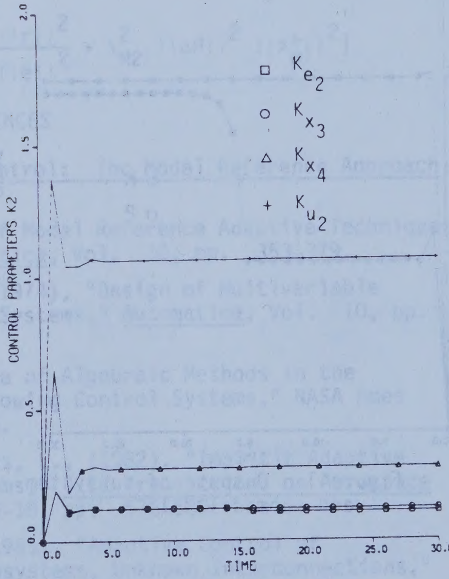


Figure 5. Control parameters for subsystem # 2

ADAPTIVE CONTROL UNDER PARTIALLY MEASURED CONTROL VARIABLES

P. Albertos and F. Morant
Dept. Ing. Sistemas Computadores y Automatica
Univ. Politecnica de Valencia
46071 Valencia
Spain

ABSTRACT

The blending problem in a cement raw materials mill is the starting point to consider a general control problem when some control inputs partially contribute to control a process, this contribution being unknown and time variable. In this paper the estimation problem is examined and a general procedure to recursively estimate the contribution of any input is developed. The procedure has been successfully applied to a cement plant.

INTRODUCTION

One of the main requirements in cement production is to obtain a good slurry composition to feed the kiln. It must be held almost constant and close to the composition computed as the optimal one. In the simplest case, that is, assuming a stable and constant operating point, an ideal blending process without delays and a known raw material composition, the weights of raw materials are obtained from a linear programming optimization problem. But, in an actual plant none of these ideal conditions hold.

First, some disturbances must be considered not allowing the assumption of constant operating conditions. So, some feedback control must be implemented, and an homogeneization silo smoothes the slurry composition at the last step in the mill plant.

Second, mill dynamics must be taken into account, introducing at least a pure dead time. These dynamics are variable mainly according to the level of loading of the mill.

Third, raw materials composition is variable and not well-known. The physical properties of these materials do not permit an easy composition analysis. Moreover, slurry quality is usually measured by means of some standards which are nonlinear functions of the slurry oxide composition.

The control variables are the raw materials weights. Usually four raw materials are handled but only weight percentages are considered for composition control. The total rate of feeding is reserved for production control.

Among the many chemical components present in raw materials only four are relevant, either because their high percentage or by their chemical properties: SiO_2 (S), CaO (C), Al_2O_3 (A) and Fe_2O_3 (F). As a percentage composition is what is interesting, three components are enough to define the slurry.

The most usual cement standards are:

[†] The author is a Visiting Professor with the Department of Electrical and Computer Engineering and a Visiting Research Professor with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. This visit was supported in part by "Conselleria de C.E. y C. de la Comunidad Autonoma Valenciana," Grant 87/1617.

$$\text{Lime standard: } LM = \frac{100C}{2.8\,S+1.1\,A+0.8F} \tag{1a}$$

$$\text{Aluminium modulus: } AM = \frac{A}{F} \tag{1b}$$

$$\text{Silica modulus: } SM = \frac{S}{A+F} \tag{1c}$$

but some others can be also used [1]. Obviously, only three at a time may be considered as controlled variables.

Then, the control problem can be stated as follows: Given a process, the mill,

$$y^{o}(z) = G_c(z)u(z) \tag{2}$$

where

- $u(z)$: vector of raw materials weights, dim 4,
- $y^{o}(z)$: vector of output oxide composition, dim 4,
- $G_c(z)$: matrix transfer function, dim 4x4,

subindex c pointing out composition parameters influence, assuming a time variable raw material composition C_k, dim 4x4, determine the controller to maintain the vector of standards, $S(z)$, at a desired set point.

The nonlinear equations (1a,b,c) can be expressed by

$$S_k = N(y_k^{o}) \tag{1}$$

where subindex k indicates a given sampling time.

The relation between raw materials weights, u_k, and oxide weights at the mill input, u_k^{o}, may be denoted by

$$u_k^{o} = C_k \cdot u_k \tag{3}$$

and a similar relation could be stated between raw materials and oxide composition output vectors, y_k and y_k^{o}, respectively, but with different matrix composition. So, the control scheme would be that shown in Figure 1.

Some rearrangements can be carried out to simplify this nonlinear time-variant multivariable problem. First, mill dynamics is modelled by the same components in the input and output vectors. So interaction is largely reduced and a new diagonal transfer matrix can be assumed. This is not a strong simplification if raw materials process is considered because the delays in the mill body are almost independent for any of the raw materials, depending mostly in their physical properties and, of course, the level of loading of the mill. That means a slight interaction. From (3), the following relationship holds:

$$G^{o}(z) = CG(z)C^{-1} \tag{4}$$

where $G(z)$ $[G^{o}(z)]$ represent the matrix transfer function when raw materials [oxides] are considered. Second, an output decoupling process may be carried out [1], [2], under

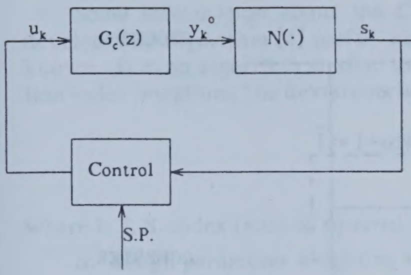


Figure 1.

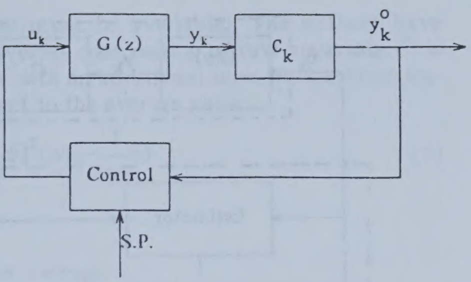


Figure 2.

some assumptions, allowing the control of each output vector component by one of the raw materials weight inputs. With these two assumptions, the control scheme may be stated as shown in Figure 2.

Some control algorithms have been applied to that scheme, usually including the silo as part of the mill. In [3] an adaptive strategy is proposed with minimum variance controllers, but assuming constant raw materials composition matrix C.

PROBLEM STATEMENT

The problem presented in this paper is mainly related to C-matrix variations in process model (Figure 3). Different assumptions about either mill dynamics or C-matrix knowledge are made:

(1) Static case with persistent excitation at inputs.

(2) Static case with average value of C-matrix elements known.

(3) Dynamics of process is considered.

(4) Same, but a model of parameters variation is known.

Figure 3 shows the main blocks in the controller. The estimator can be developed to estimate either the elements of C or C^{-1} matrices. Nevertheless, from a user point of view those of C-matrix are much more interesting. As later stated (5), only some elements of C-matrix are variable and nonzero, so the inverse matrix $\hat{C}^{-1}$ can be easily computed. The transfer matrix G(z) will be considered as a diagonal matrix, its elements being constant in cases 1 and 2.

Questions related to the control algorithms will not be considered here as they can be separated from those of parameter estimation procedure. Nevertheless, questions about global system stability can arise and clearly depend on the control and estimation algorithms used.

In a general case, the C-matrix must be considered a full-matrix. In the cement application some assumptions are plausible:

-Sand is mostly silica and its composition is known.

-Pyrites are mainly Fe_2O_3, and its composition is also known.

So, the following approximation is valid:

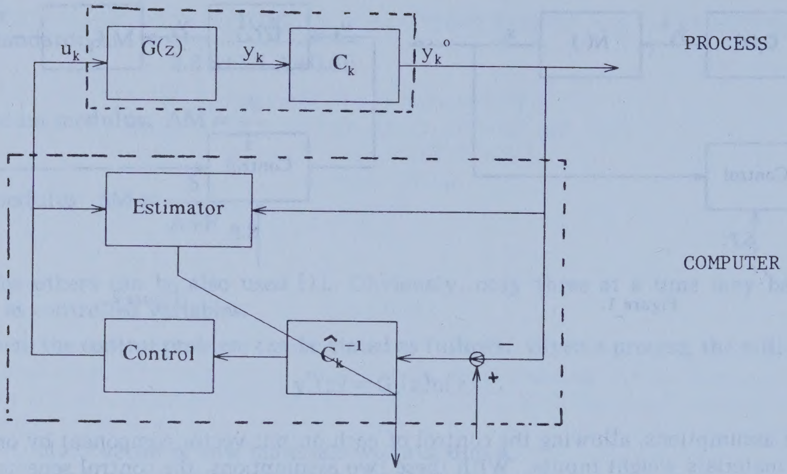


Figure 3.

$$\begin{vmatrix} S \\ F \\ C \\ A \end{vmatrix} = \begin{vmatrix} SS & 0 & SC & SL \\ 0 & FP & FC & FL \\ 0 & 0 & CC & CL \\ 0 & 0 & AC & AL \end{vmatrix} \begin{vmatrix} Sa \\ P \\ Cl \\ L \end{vmatrix} \tag{5}$$

$$O = C \cdot R$$

This expression relates oxides, O, and raw materials, R. The inverse relation is similar, with the same nonzero elements, and so, to get C^{-1} a very short computational time is required.

STATIC SOLUTION

(1) The solution is straightforward in the first case. Four different estimation problems can be stated, one for each oxide. From different samples a measurement vector is obtained with raw materials weights as components and a measurement matrix is formed. Note that the same matrix is valid for any oxide. Then, by some least-square (L.S.) parameter estimation algorithm, elements of C-matrix can be estimated. To simplify the notation, a unit gain is assumed in the mill. So for any of the oxide outputs we can express

$$y_{i,k}^{o} = \sum_{j=1}^{4} c_{ij} u_{j,k} \,. \tag{6}$$

The vector of parameters $\theta_i = [C_{i1}, \ldots, C_{i4}]^T$ is estimated from the least N vector of inputs and outputs, usually in a recursive way.

(2) The problem is quite different if, as usual, the process works under normal operating conditions and so the inputs remain almost invariant. Persistent excitation condition is lost and the algorithm above used becomes unstable.

Some information about the C-elements must be available. The authors have developed an algorithm [4] useful when the average composition of raw materials, $\bar{C}$, is known. It is an algorithm similar to L.S. one with an additional term in the minimization index weighting the deviations with respect to the average value.

$$\tilde{I} = I+\alpha N(M_N-\frac{1}{N}\theta)^T(M_N-\frac{1}{N}\theta) \tag{7}$$

where I: L.S. index (sum of squared errors)

α: design parameter weighting the known average,

$M_N = \bar{\theta}-\frac{1}{N}\sum_{r=1}^{N-1}\hat{\theta}_r$

$\bar{\theta}$: parameters average,

$\hat{\theta}_r$: estimated θ parameters after r samples.

As before, the same measurement matrix is valid for the estimation of the parameters related to any oxide (elements of a row in the C-matrix).

PROCESS DYNAMICS

Let us consider case 3, that is, process delays are included. For any of the elements of the diagonal matrix G(z), a rational z-transfer function

$$G_i(z) = B_i(z)\cdot z^{-d_i}/A_i(z) \tag{8}$$

is assumed. Usually a few parameters are necessary to sufficiently model the storage process in the mill. First, changes in C-parameters were assumed to be slower than those of the mill. So they can be considered constant in the identification algorithm.

As only u_k and $\overset{o}{y}_k$ can be measured, a computation of y_k is performed by means of the process model, $\tilde{y}(z) = G(z)u(z)$, Figure 3. The preceding estimation algorithm is applied now with a measurement matrix built with $\tilde{y}_k$ and $\overset{o}{y}_k$, instead of u_k and $\overset{o}{y}_k$, (6).

When the C-constancy assumption is not valid, then raw materials composition must be taken into account in the mill dynamics. It will be easy if a pure time delay d is assumed to model the mill. Then from $\tilde{y}_k$ and $\overset{o}{y}_k$ the value of C_{ii} elements can be estimated for d instants of time before, $\tilde{y}_k$ being computed from u_{k-d}.

Formally, in that case, the process model becomes nonlinear. We can transform it to a linear one if we consider as input variables $\overset{o}{u}_k = C_k\cdot u_k$, input oxides in the cement application. The estimation problem may now be split into two sequential problems: (1) a deconvolution problem to estimate $\overset{o}{u}_k$ from output measurements, assuming the process model is known and (2) the computation sequence $\{C_k\}$ from inputs u_k and the estimated of $\overset{o}{u}_k$.

The deconvolution problem is solved using a Schmidt-Kalman filter [5]. So, process (8) is expressed in state space form as

$$x_{k+1} = Ax_k+B\overset{o}{u}_k+v_{1,k} \tag{9a}$$

$$y_k^o = Dx_k + v_{2,k} , \qquad \textbf{(9b)}$$

where (A,B,D) are the model matrices and $v_{1,k}$, $v_{2,k}$ two independent zero mean random processes with known covariance matrices.

A piecewise constant model is used for the input:

$$u_{k+1}^o = u_k^o + \eta_k . \qquad (10)$$

The augmented state vector, $[x_k^T, u_k^{oT}]^T$ is estimated and then, from a set of equations similar to (3) the parameters are estimated [6].

It must be pointed out that, as mentioned at the beginning, when oxides are considered in the model, (9) is more interactive than when raw materials are considered. This is an additional difficulty to deal with at the control design stage.

As a delay must always be considered, an additional problem arises. What we need for control purposes is the value of the matrix composition just at the present time and the estimation algorithm gives us C_i at the input di periods of time before. Then, let us consider part 4 of our problem. If a model of the mixing-beds are available, we can use it as a C_i-predictor at time k, from the values previously estimated.

The above procedure has been successfully applied to a cement plant in real time. In the estimation phase, only pure delay time in the mill was assumed in spite of a more detailed model for control purposes, [1].

CONCLUSION

An attempt has been made to formalize the problem of input parameters estimation in the blending process. Different scenarios have been considered and a solution is proposed in any case. Obviously when blending dynamics are considered the problem is more complex and a previous deconvolution algorithm must be performed.

REFERENCES

[1] Morant, F. Albertos, P. and Crespo, P., "Oxide composition control in a raw material mill," *IFAC Symp. on Microprocessors in Control*, Istantbul, 1986.

[2] Keviczky, L., et al., "A microprocessor based adaptive composition control system," *IFAC Symp. on Real-Time Control Applications*, Guadalajara, Mexico, 1983.

[3] Keviczky, L., "Self-tuning adaptive control of cement raw material blending," *Automatica*, Vol. 14, pp. 525-532, 1978.

[4] Albertos, P. and Morant, F., "An algorithm for parameter estimation with multiple undefined solutions: the blending problem," *IFAC/IFORS Symp. on Identification and System Parameter Estimation*, York, 1985.

[5] Candy, J. V., *Signal Processing, The Model-Based Approach*, McGraw-Hill, 1986.

[6] Jazwinski, A. H., *Stochastic Processes and Filtering Theory*, Academic Press, 1970.

Disturbance-generated bifurcation in a prototype adaptive system with e_1– modification law

Shi BAI and Fathi M.A. SALAM

Department of Electrical Engineering and Systems Science,
Michigan State University,
East Lansing, MI 48823, USA

ABSTRACT

A new modified adaptation law, the so-called e_1–modification law, has been proposed by Narendra and Annaswamy (1987) for robust model reference adaptive control in the presence of bounded disturbances. Using this law, we show that in the case of regulation a prototype adaptive control system, with a single unknown pole and an unknown constant disturbance at its input, possesses multiple equilibria and undergoes various bifurcations. Our simulations and local analyses give a complete characterization of the possible dynamics exhibited by the disturbed adaptive system. As the constant disturbance is varied, saddle-node bifurcation, (subcritical) Hopf bifurcation, and saddle-connection bifurcation occur in order of precedence.

Keywords: Simulation, Adaptive control systems, e_1–modification law, Stability, Saddle-node bifurcation, Hopf bifurcation, Saddle-connection bifurcation.

1. Introduction

The so-called e_1–modification adaptation law has recently been proposed by Narendra and Annaswamy [1] for model reference adaptive control systems in the presence of bounded external disturbances and unmodelled dynamics. It has been shown to improve certain characteristics of the adaptive system while ensuring boundedness of all solutions, see [1] and the references therein.

We are interested in a detailed study of the performance of the e_1–modification law when applied to a first order plant with an unknown pole and a bounded external disturbance. The disturbance considered has a constant average value, say d, and small additive fluctuating term, say $\delta(t)$. That is, the external disturbance has the form:

External disturbance = $d + \delta(t)$,
where $|\delta(t)| \ll 1$.

In this work, we limit attention to the case when the fluctuation, $\delta(t)$, equals zero. Consequently, we focus on determining the properties of the whole adaptive system in the presence of the constant disturbance d. Available methods in analysis can consequently be used to determine the corresponding properties of the whole system when the fluctuating disturbance $\delta(t) \neq 0$. In what follows we show that all, necessarily bounded, solutions can in fact converge to or diverge from every possible limit set in the two dimensional plane, namely, an equilibrium point, a limit cycle or a homoclinic orbit. The type of limit set present in the system is dictated by the relative values of all (constant) parameters, i.e., the constant unknown disturbance, the amplitude of e_1–modification term, etc.. Specifically, we present simulations and some local analyses that completely characterize all possible dynamical behavior of the prototype adaptive control system distinguished by the e_1–modification law. We have determined the number, the location, and the local stability of the equilibria; we have also derived explicit expressions for saddle-node and Hopf bifurcations. Based on our simulations and the local analyses, three types of bifurcations have been identified:

(1) saddle-node bifurcation
(2) (subcritical) Hopf bifurcation
(3) saddle-connection bifurcation

The presentation is organized as follows. In section 2, we introduce the prototype adaptive control system frequently used in the recent literature on adaptive control systems. Then, we determine the number and location of the equilibria. In section 3, we analyse the local stability of the equilibria and present the bifurcation expressions. In section 4, we explain the simulation results for different constant disturbances (while all other parameters are fixed). In section 5, we list some conclusions.

2. Adaptive Control of A First-Order Plant

In the prototype adaptive control system the (first-order) plant is described by the scalar differential equation (see [1], [3] and [14])

$$\dot{x}_p = a_p x_p + u + d, \tag{1}$$

where a_p is an unknown constant, d is an unknown disturbance, and u is the control. Assume the disturbance to be constant. The reference model is described by the equation

$$\dot{x}_m = - a_m x_m + r, \tag{2}$$

where $a_m > 0$ and r is a bounded piecewise continuous function. The adaptive controller is usually chosen in the standard form as

$$u = r + \theta x_p, \tag{3}$$

where the feedback control parameter θ is adjusted via an adaptation law that uses the available input-output data.

Now let the state (or output) error be

$$e_1 = x_p - x_m \,,$$

then the adaptation law proposed in [1] reads as:

$$\dot{\theta} = - e_1 x_p - \gamma | e_1 | \theta \,, \ \gamma > 0. \tag{4}$$

The term ($-\gamma | e_1 | \theta$) has been proposed and referred to as the e_1–modification scheme, see Narendra and Annaswamy [1].

Using equations (1) - (4), one derives the equivalent system:

$$\dot{e}_1 = a_p e_1 + \theta e_1 + d + \theta x_m + (a_m + a_p) x_m, \tag{5}$$

$$\dot{\theta} = - e_1^2 - \gamma | e_1 | \theta - e_1 x_m, \tag{6}$$

$$\dot{x}_m = - a_m x_m + r. \tag{7}$$

For now consider the regulation problem, i.e., let $r = 0$. The equilibria then must satisfy

$$0 = a_p e_1 + \theta e_1 + d, \tag{8i}$$

$$0 = - e_1^2 - \gamma | e_1 | \theta, \tag{8ii}$$

$$0 = x_m. \tag{8iii}$$

Write

$$| e_1 | = e_1 sgn(e_1), \tag{9}$$

where sgn is the sign function. Then using (8i), $sgn(e_1)$ and (8ii), one obtains the quadratic polynomial in e_1

$$e_1^2 - \gamma sgn(e_1)[a_p e_1 + d] = 0. \tag{10}$$

Thus, one obtains an explicit expression for the e_1–component of the equilibria as

$$e_1 = \frac{1}{2}\gamma a_p sgn(e_1)\left[1 \pm \sqrt{1 + \frac{4d sgn(e_1)}{\gamma a_p^2}}\,\right]. \tag{11}$$

From Eqn.(11), one may write a 'bifurcation' expression for the number of equilibria as

$$d_0 = \frac{1}{4}\gamma a_p^{2}. \tag{12}$$

Thus the equilibria are (see Appendix A):
(1) for $| d | > d_0$, there is a single equilibrium point.
(2) at $| d | = d_0$, there are three equilibria, two of which are equal.
(3) for $| d | < d_0$, there are three distinct equilibria.
There is always one, and only one, equilibrium point with positive (respectively, negative) e_1 component for all positive (respectively, negative) values of d.

A REMARK:
The σ–modification adaptation law is given by

$$\dot{\theta} = -e_1 x_p - \sigma\theta,$$

where σ is a positive constant, i.e., $\sigma > 0$. The new modified adaptation law, proposed in [1], basically replaces σ by $\gamma|e_1|$.

3. Local Stability Analysis

As it is often the case, see [3] and [14], the initial point of the variable x_m, i.e. $x_m(0)$, shall be set to zero; consequently x_m (t) = 0 for all t. In essence, we are considering the simpler system:

$$\dot{e}_1 = a_p e_1 + \theta e_1 + d, \tag{13}$$

$$\dot{\theta} = -e_1^{2} - \gamma|e_1|\theta. \tag{14}$$

This system captures the essential dynamic features of the adaptive control system (1)-(4). The dynamic features are dependent on the relative values that (the system paramenter) a_p, (the adaptive control parameter) γ, and (the external disturbance parameter) d will take. Expression (12) exhibits the dependency of the number of equilibria on the relative values of these parameters.

Now we analyse the stability of each equilibrium point. The linearized system at any equilibrium point reads

$$\begin{bmatrix} \dot{\tilde{e}}_1 \\ \dot{\tilde{\theta}} \end{bmatrix} = \begin{bmatrix} a_p + \overline{\theta} & \overline{e}_1 \\ -2\overline{e}_1 - \gamma sgn(\overline{e}_1)\overline{\theta} & -\gamma sgn(\overline{e}_1)\overline{e}_1 \end{bmatrix} \begin{bmatrix} \tilde{e}_1 \\ \tilde{\theta} \end{bmatrix}. \tag{15}$$

Let us denote the determinant of the Jacobian matrix by det. Then,

$$\det = 2\overline{e}_1^{2} - a_p\gamma sgn(\overline{e}_1)\overline{e}_1$$

$$= 2\overline{e}_1[\overline{e}_1 - \frac{1}{2}\gamma a_p sgn(\overline{e}_1)].$$

Let us also denote the trace of the Jacobian matrix by Trace. Then,

$$Trace = a_p + \overline{\theta} - \gamma sgn(\overline{e}_1)\overline{e}_1$$

$$= -\frac{1}{\overline{e}_1}[d + \gamma sgn(\overline{e}_1)\overline{e}_1^{2}].$$

To ease the presentation, the following discussion only considers the case $d > 0$; a similar conclusion follows for $d < 0$.

(i) For $d > d_0$ (There exists a single equilibrium point only.)

$$\det > 0,$$

$$Trace < 0,$$

so the two eigenvalues are both negative. Thus the equilibrium point is (locally) stable.

(ii) For $d = d_0$ (There exist two distinct equilibrium points.) First,

$$[\overline{e}_1]_1 = \frac{1}{2}\gamma a_p(1 + \sqrt{2}),$$

$$\det > 0,$$

$$\mathit{Trace} < 0,$$

so the two eigenvalues are negative. Thus this equilibrium point is (locally) stable.
Now considering the coincident two equilibria, one has

$$[\bar{e}_1]_{2,3} = -\frac{1}{2}\gamma a_p,$$

$$\det = 0,$$

$$\mathit{Trace} = \frac{1}{2}a_p(1 - \gamma^2),$$

so there is a zero eigenvalue. Thus the coincident equilibria are degenerate.

(iii) For $d < d_0$ (There are three distinct equilibria.)
First,

$$[\bar{e}_1]_1 = \frac{1}{2}\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^{\,2}}}\,\right],$$

$$\det > 0,$$

$$\mathit{Trace} < 0,$$

so that the two eigenvalues are negative. Thus, this equilibrium point is (locally) stable.
The second equilibrium point gives,

$$[\bar{e}_1]_2 = -\frac{1}{2}\gamma a_p\left[1 - \sqrt{1 - \frac{4d}{\gamma a_p^{\,2}}}\,\right],$$

$$\det < 0,$$

$$\mathit{Trace} \neq 0,$$

so one eigenvalue is positive and another is negative. Thus $([\bar{e}_1]_2,[\bar{\theta}]_2)$ is a saddle point. The third equilibrium gives,

$$[\bar{e}_1]_3 = -\frac{1}{2}\gamma a_p\left[1 + \sqrt{1 - \frac{4d}{\gamma a_p^{\,2}}}\,\right],$$

$$\det > 0,$$

$$\mathit{Trace} = 0 \text{ if}$$

$$d - \frac{1}{4}\gamma^3 a_p^{\,2}\left[1 + \sqrt{1 - \frac{4d}{\gamma a_p^{\,2}}}\,\right]^2 = 0.$$

One may write a 'Hopf bifurcation' expression for the critical stability of this equilibrium point as

$$d_H = \frac{a_p^{\,2}\gamma^3}{(1 + \gamma^2)^2}. \tag{16}$$

Consequently, we have

$\mathit{Trace} < 0$, for $d < d_H$, and thus this equilibrium point is stable
$\mathit{Trace} = 0$, for $d = d_H$, and thus this equilibrium point is critical
$\mathit{Trace} > 0$, for $d > d_H$, and thus this equilibrium point is unstable

at $d = d_H$, we also have $\left.\dfrac{\partial(\mathit{Trace})}{\partial d}\right|_{d_H} \neq 0.$

One immediate conclusion is that the stability has changed at this equilibrium point when the value of d passes through the value of d_H. In addition a (subcritical) Hopf bifurcation takes place at $d = d_H$. Recall that at $d = d_0$ a saddle-node bifurcation occurs also.

4. Simulation Results

We have considered the system (5)-(7) with $x_m(0) = 0$, in other word, we have considered the simpler two-dimensional system (13)-(14), and have concentrated on the invariant curves in the space of e_1 and θ. Specifically, we have concentrated on the stable and unstable manifolds of the saddle point when $|d| < d_0$. The parameters have been fixed as follows(recall that $r = 0$):

$$a_p = 4.0\ , \quad \gamma = 0.5$$

Therefore, the equilibria-bifurcation value of d now reads

$$d_0 = \frac{1}{4}\gamma a_p{}^2 = 2.0$$

Hence for $|d| < 2.0$, there exist three equilibria. We have chosen the disturbance variable d as the varying parameter. For simplicity of the presentation we only discuss the simulation results as the disturbance d increases from 0 to 2.5. We now discuss the simulation results.

At $d = 0.2$, Figure 1 displays three equilibrium points: two stable equilibrium points and one saddle lying near $(e_1 , \theta) = (- 0.05132,- 0.10263)$. The four initial points are located along the two eigenvectors of the linearization of the two dimensional system at the saddle point evolving forward or backward in time and thus trace the stable and unstable manifolds of the saddle. The arrows indicate the direction of the trajectories. Observe that the unstable manifold of the saddle is engulfed by its own stable manifold which in turn converges to the 'left' stable equilibrium point. Although the (local) stability properties of the equilibrium points remain the same for $0 \le d < 1.28$, but the global qualitative behavior persists for only $0 < d < 0.43794904$.

At $d = 0.43794904$, Figure 2 shows that the branches of the stable and unstable manifolds are coincident and engulf the 'left' stable equilibrium point, forming the so-called homoclinic orbit. The homoclinic orbit defines a structure of considerable interest in dynamical systems since it precedes the rise of 'horseshoe chaos'. This orbit is the coincidence of the stable and unstable manifolds of the saddle and has the property that each point on the structure would converge to the saddle point when evolving forward or backward in time.

At $d = 1.0$, Figure 3 shows that a new α–limit set now appears: an (unstable) limit cycle. The equilibrium point inside the limit cycle is (locally) asymptotic stable. As d increases the limit cycle decreases in amplitude and the attraction of the equilibrium point inside it weakens.

At $d = 1.5$, Figure 4 shows the limit cycle and the equilibrium point it is engulfing have fused into one another forming a locally unstable equilibrium point (which is a source).

The newly created source and the saddle have continued to approach one another and at $d = d_0 = 2.0$ have fused into each orther forming saddle-node bifurcation. Figure 5 shows the manifolds connected to the now fused equilibrium. The linearized system now possess a zero eigenvalue at this fused equilibrium point [9].

When $d > 2.0$, the two fused equilibria wipe out one another and the system now possesses a single stable equilibrium point with positive e_1 component. The asymptotic stability of this equilibrium point will remain the same for all values of $d > 2.0$. Figure 6 shows some trajectories converging to the single stable equilibrium at $d = 2.5$. The further discussion for fixes d and varies γ, consult [14] and the similar regults will be obtained.

5. Conclusions

We summarize further conclusions in the following:

1. The e_1–modification scheme generates very rich dynamics in the presence of external (constant) disturbance in the regulation case, i.e. $r = 0$.
2. The system exhibits rich dynamics and structures. Specifically, as the disturbance parameter d is decreased from 2.5 to 0, the system consecutively experiences saddle-node, Hopf, and saddle connection bifurcations. At $d = 0.43794904$, a homoclinic orbit exists.
3. The simulation results however illustrate that the 'left' equilibrium point changes stability. For $0 \le d < 1.28$ ($= d_H$), this equilibrium point is stable; however, for $1.28 < d < 2.0$ ($= d_0$), it is unstable.
4. The presence of the homoclinic orbit implies the possibility that the addition of disturbance, which is a small sinusoidal function of time, could lead to the generation of a (transversal) homoclinic orbit. Such an occurrence implies the existence of horseshoes in the dynamics of the system, thus giving rise to the

so-called horseshoe chaos.

5. Indeed, the e_1–modification scheme exhibits qualitatively the same rich dynamics as the σ–modification scheme, as shown in [14]. So, in as far as dealing with constant disturbances, both schemes perform similarly.

6. Appendix A

The equilibria are:

(1) for $| d | > d_0$, there is a single equilibrium point, namely,

$$[\bar{e}_1] = \begin{cases} \frac{1}{2}\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p{}^2}}\ \right], & d > d_0 \\ -\frac{1}{2}\gamma a_p\left[1 + \sqrt{1 - \frac{4d}{\gamma a_p{}^2}}\ \right], & d < -d_0 \end{cases}$$

$$[\bar{\theta}] = -a_p - \frac{d}{[\bar{e}_1]}.$$

(2) at $| d | = d_0$, there are three equilibria, two of which are equal, namely,

(2i) when $d = d_0$

$$[\bar{e}_1]_1 = \frac{1}{2}\gamma a_p[1 + \sqrt{2}],$$

$$[\bar{e}_1]_{2,3} = -\frac{1}{2}\gamma a_p,$$

$$[\bar{\theta}]_{1,2,3} = -a_p - \frac{d}{[\bar{e}_1]_{1,2,3}}.$$

(2ii) when $d = -d_0$

$$[\bar{e}_1]_1 = -\frac{1}{2}\gamma a_p[1 + \sqrt{2}],$$

$$[\bar{e}_1]_{2,3} = \frac{1}{2}\gamma a_p,$$

$$[\bar{\theta}]_{1,2,3} = -a_p - \frac{d}{[\bar{e}_1]_{1,2,3}}.$$

(3) for $| d | < d_0$, there are three distinct equilibria, namely,

(3i) when $0 \leq d < d_0$

$$[\bar{e}_1]_1 = \frac{1}{2}\gamma a_p\left[\ 1 + \sqrt{1 + \frac{4d}{\gamma a_p{}^2}}\ \right],$$

$$[\bar{e}_1]_2 = -\frac{1}{2}\gamma a_p\left[\ 1 - \sqrt{1 - \frac{4d}{\gamma a_p{}^2}}\ \right],$$

$$[\bar{e}_1]_3 = -\frac{1}{2}\gamma a_p\left[\ 1 + \sqrt{1 - \frac{4d}{\gamma a_p{}^2}}\ \right],$$

$$[\bar{\theta}]_{1,2,3} = -a_p - \frac{d}{[\bar{e}_1]_{1,2,3}}.$$

(3ii) when $-d_0 < d \leq 0$

$$[\bar{e}_1]_1 = -\frac{1}{2}\gamma a_p\left[\ 1 + \sqrt{1 - \frac{4d}{\gamma a_p{}^2}}\ \right],$$

$$[\bar{e}_1]_2 = \frac{1}{2}\gamma a_p\left[\ 1 - \sqrt{1 + \frac{4d}{\gamma a_p{}^2}}\ \right],$$

$$[\bar{e}_1]_3 = \frac{1}{2}\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right],$$

$$[\theta]_{1,2,3} = -a_p - \frac{d}{[\bar{e}_1]_{1,2,3}}.$$

7. Appendix B

(i) For $d > d_0$ (There exists a single equilibrium point only.)

$$\det = \frac{1}{2}\gamma^2 a_p^2\sqrt{1 + \frac{4d}{\gamma a_p^2}}\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right] > 0,$$

$$Trace = -\frac{2d}{\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right]} - \frac{1}{2}\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right] < 0,$$

so the two eigenvalues are both negative. Thus the equilibrium point is (locally) stable.

(ii) For $d = d_0$ (There exist two distinct equilibrium points.) First,

$$[\bar{e}_1]_1 = \frac{1}{2}\gamma a_p(1 + \sqrt{2}),$$

$$\det = \gamma^2 a_p^2\left[1 + \frac{\sqrt{2}}{2}\right],$$

$$Trace = -\frac{2d}{\gamma a_p(1 + \sqrt{2})} - \frac{1}{2}\gamma^2 a_p(1 + \sqrt{2}),$$

so the two eigenvalues are negative. Thus this equilibrium point is (locally) stable. Now considering the coincident two equilibria, one has

$$[\bar{e}_1]_{2,3} = -\frac{1}{2}\gamma a_p,$$

$$\det = 0,$$

$$Trace = \frac{1}{2}a_p(1 - \gamma^2),$$

hence, there is a zero eigenvalue. Thus the coincident equilibria are degenerate.

(iii) For $d < d_0$ (There are three distinct equilibria.)

First,

$$[\bar{e}_1]_1 = \frac{1}{2}\gamma a_p\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right],$$

$$\det = \frac{1}{2}\gamma^2 a_p^2\sqrt{1 + \frac{4d}{\gamma a_p^2}}\left[1 + \sqrt{1 + \frac{4d}{\gamma a_p^2}}\right],$$

$$Trace = -\frac{2d}{\gamma a_p\left[1 + \sqrt{1 - \frac{4d}{\gamma a_p^2}}\right]} - \frac{1}{2}\gamma^2 a_p\left[1 + \sqrt{1 - \frac{4d}{\gamma a_p^2}}\right],$$

so that the two eigenvalues are negative. Thus, this equilibrium point is (locally) stable. The second equilibrium point gives,

$$[\bar{e}_1]_2 = -\frac{1}{2}\gamma a_p\left[1 - \sqrt{1 - \frac{4d}{\gamma a_p^2}}\right],$$

$$\det = -\frac{1}{2}\gamma^2 a_p^2\sqrt{1 - \frac{4d}{\gamma a_p^2}}\left[1 - \sqrt{1 - \frac{4d}{\gamma a_p^2}}\right],$$

$$Trace = \frac{2d}{\gamma a_p\left[1-\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right]} - \frac{1}{2}\gamma^2 a_p\left[1-\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right],$$

so one eigenvalue is positive and another is negative. Thus $([\bar{e}_1]_2,[\bar{\theta}]_2)$ is a saddle point. The third equilibrium gives,

$$[\bar{e}_1]_3 = -\frac{1}{2}\gamma a_p\left[1+\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right],$$

$$\det = \frac{1}{2}\gamma^2 a_p{}^2\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\left[1+\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right],$$

$$Trace = \frac{1}{\frac{1}{2}\gamma a_p\left[1+\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right]}\left[d-\frac{1}{4}\gamma^3 a_p{}^2\left[1+\sqrt{1-\frac{4d}{\gamma a_p{}^2}}\right]^2\right].$$

Acknowledgement:

Research is supported by NSF under grant ECS–8702889.

References

[1] K.S. Narendra and A.M. Annaswamy, A new adaptive law for robust adaptive without persistent excitation, *IEEE Trans. Automat. Control.* **32** *(1987) 134-145.*

[2] K.S. Narendra and A.M, Annaswamy, A general approach to the stability analysis of adaptive systems, *Proceedings of The 23th IEEE Conference on Decision and Control.* Las Vegas, NV (December 1984) 1298-1303.

[3] B.Riedle, B. Cyr and P.V. Kokotovic, Disturbance instability in an adaptive system, *IEEE Trans. Automat. Control.* **29** *(1984) 822-824.*

[4] P.I. Ioannou and P.V. Kokotovic, Instability analysis and robustness of adaptive control, *Automatica* **20** *(1984).*

[5] C.E. Rohrs, L. Valavani, M. Athans and G. Stein, Robustness of continuous-time adaptive control algorithms in the presence of unmodelled dynamics, *IEEE Trans. Automat. Control* **30** *(1985) 881-889.*

[6] M. Bodson and S. Sastry, Exponential convergence and robustness margins in adaptive control, *Proceedings of The 23th IEEE Conference on Decision and Control,* Las Vegas, NV (December 1984) 1282-1285.

[7] S. Sastry, Model reference adaptive control - stability, parameter convergence, and robustness, *IMA J. Math. Control Inform.* **1** *(1984) 27-66*

[8] S. Boyd and S. Sastry, Necessary and sufficient conditions for parameter convergence in adaptive control, *proc. of the Berkeley-Ames Conference on Nonlinear Problems in Control and Fluid Dynamics* (Math. Sci. Press, Brookline, MA, 1984) 81-101.

[9] S.N. Chow and J.K. Hale, *Methods of Bifurcation Theory* (Springer, Berlin-New York, 1982).

[10] J. Carr, *Applications of Centre Manifold Theory* (Springer, Berlin-New York, 1981).

[11] B.D. Hassard, N.D. Kazarinoff and Y.-H. Wan, *Theory and Applications of Hopf Bifurcation* (Cambridge Univ. Press, Cambridge, 1981).

[12] J.E. Marsden and M. McCracken, *The Hopf Bifurcation and Its Applications* (Springer, Berlin-New York, 1976).

[13] F.M.A. Salam, Parameter space analysis and design of an adaptive system, *Proceedings of The 25th IEEE Conference on Decision and Control,* (December 1986) 1155-1160

[14] F.M.A. Salam and S. Bai, Disturbance-generated bifurcations in a simple adaptive system: Simulation evidence, *Systems Control Lett.* **7** *(1986) 269-280.*

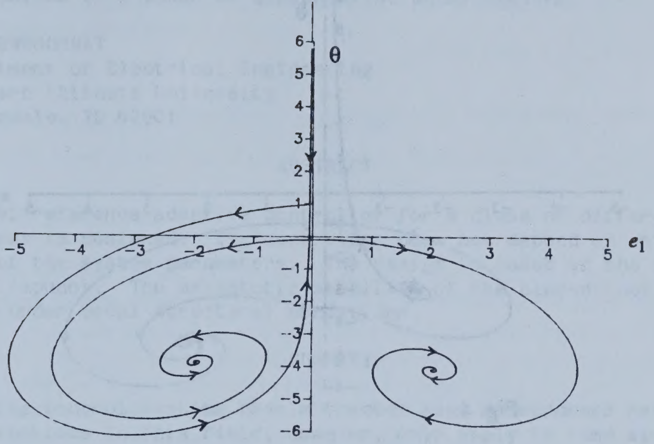


Fig. 1

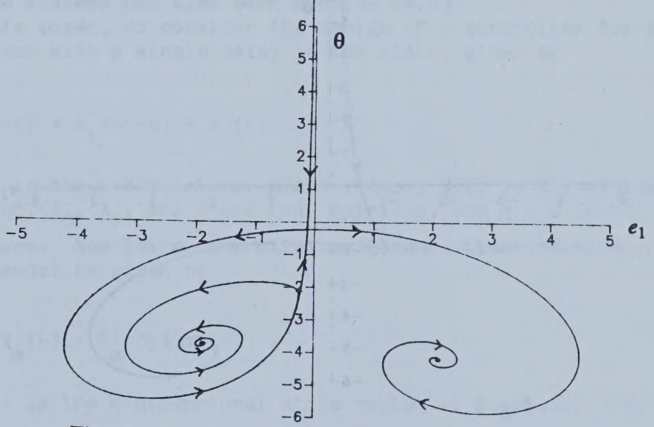


Fig. 2

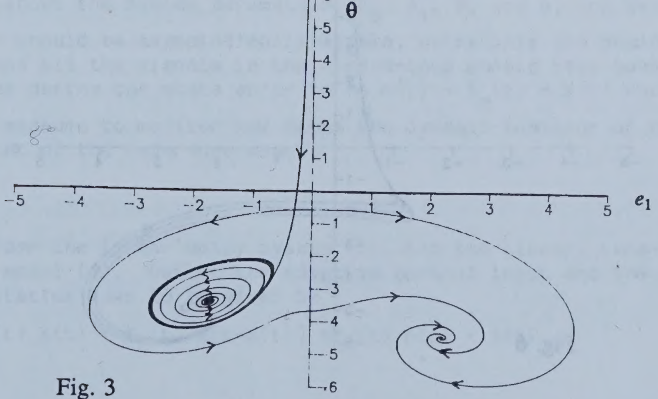


Fig. 3

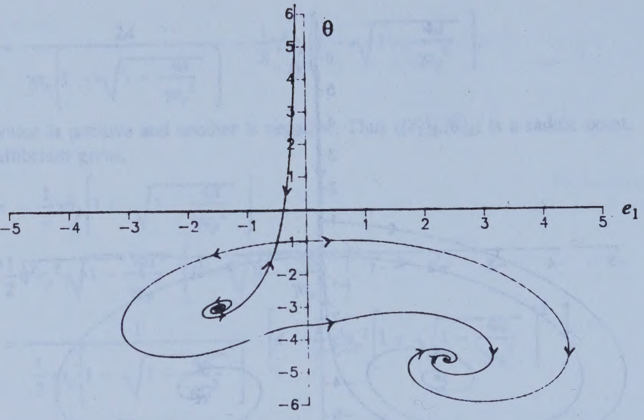


Fig. 4

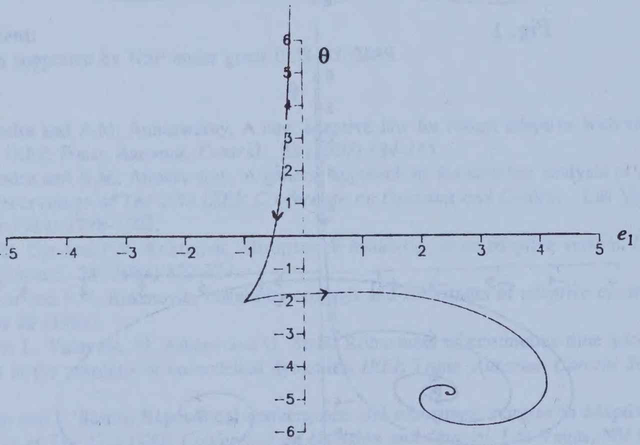


Fig. 5

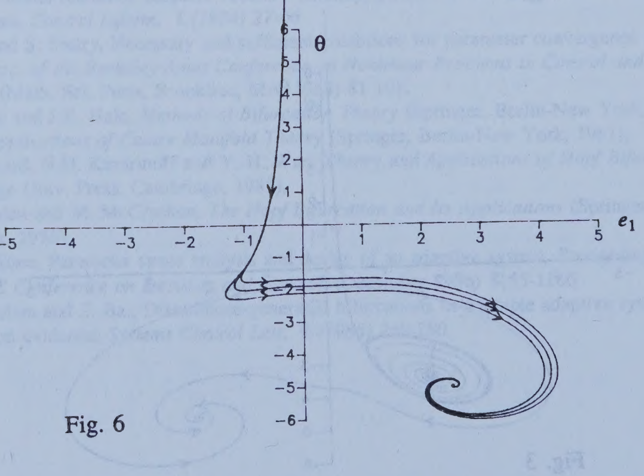


Fig. 6

ADAPTIVE CONTROL OF A CLASS OF DIFFERENTIAL DELAY SYSTEMS

F. POURBOGHRAT
Department of Electrical Engineering
Southern Illinois University
Carbondale, IL 62901

ABSTRACT

A model reference adaptive controller for a class of differential delay systems is designed. The controller does not depend on any prior knowledge of the system parameters. The design is based on the direct method of Liapunov. The asymptotic stability of the closed-loop system is guaranteed under usual structural conditions.

SUMMARY

Adaptive control systems have attracted many researchers recently. Most contributions in this field, however, only apply to some special class of linear time-invariant systems [1,2,3]. The adaptive control of more generalized systems has also been studied [4,5].

In this paper, we consider the design of a controller for a class of delay systems with a single delay in the state, given by

$$\dot{X}(t) = A_0 X(t) + A_1 X(t-h) + BU(t) \tag{1}$$

where $X(t)$ is the n-dimensional state vector, $U(t)$ is the m-dimensional input vector, A_0, A_1, and B are real matrices, and $h > 0$ is the time delay of the system. Now let a controllable, linear, time-invariant, and stable reference model be given by

$$\dot{X}_m(t) = A_m X_m(t) + B_m r(t) \tag{2}$$

where $X_m(t)$ is the n-dimensional state vector of the model, $r(t)$ is a known m-dimensional reference input, and A_m and B_m are the model's matrices.

The problem is to design an adaptive controller so that the dynamic behavior of the system follows that of the reference model, where no prior knowledge about the system parameters, A_0, A_1, B, and h, are available. The design should be asymptotically stable, using only the measurable signals, and all the signals in the closed-loop should stay bounded.

Let us define the state error to be $e(t) = X_m(t) - X(t)$ which can be used as a measure to monitor how close the dynamic behavior of the system follows that of the reference model.

Result

Consider the linear delay system (1), and the linear, time-invariant reference model (2). Define the adaptive control input and the controller gains adaptation laws to be given by

$$U(t) = K_0(t)\, X(t) + K_1(t)\, X[t-d(t)] + K_r(t)\, r(t) + Z(t) \tag{3a}$$

and

$$\dot{K}_0(t) = \gamma_0\ J'\ P\ e(t)\ X'(t) \qquad (3b)$$

$$\dot{K}_1(t) = \gamma_1\ J'\ P\ e(t)\ X'[t-d(t)]$$

$$\dot{K}_r(t) = \gamma_r\ J'\ P\ e(t)\ r'(t)$$

$$\dot{Z}(t) = \gamma_g\ J'\ P\ e(t)$$

$$\dot{d}(t) = \gamma_h\ \mathrm{tr}\{\ J\ J'\ P\ e(t)\ \big(\ A_m X_m[t-d(t)] + B_m r[t-d(t)]]\big)'\}$$

where prime means the transpose, "tr" means the trace, γ_0, γ_1, γ_r, γ_g, and γ_h are positive real numbers, $J' = [0,I]$, and the matrix P is a positive definite symmetric matrix such that $P\ A_m + A_m'\ P = -Q$, and Q is a positive definite symmetric matrix. Then the above adaptive control input U(t) forces the dynamic behavior of the delay system (1) to follow that of the linear model (2). Moreover, it guarantees the asymptotic stability of the closed-loop system. In other words, the state error approaches zero asymptotically.

Sketch of the proof

First find the dynamic equation of the state error by subtracting equation (1) from (2), and plugging for U(t) from equation (3a). Next, let the Liapunóv function candidate V(t) be quadratic in the state error and all the parameter errors, which is a positive definite function. Take the derivative of the Liapunov function along the error dynamic equation, and combine it with the controller gains adaptation laws (3b). It can be shown that if the rate of adaptation is much higher than the system's dynamic variations, and under some usual structural conditions, then $\dot{V}(t)$ is a negative definite function which guarantees the asymptotic stability of the closed-loop system. □

References

[1] - R.V. Monopoli, "Model Reference Adaptive Control with An Augmented Error Signal," IEEE Trans. Autom. Cont., Vol. AC-19, pp. 474-484, Oct. 1974.

[2] - K.S. Narendra and L.S. Valavani, "Stable Adaptive Controller Design - Direct Control," IEEE Trans. Autom. Cont., Vol. AC-23, pp. 570-583, Aug. 1978.

[3] - D.L. Mattern and R. Shoureshi, "Model Reference Adaptive Control - A Perspective," Proc., Amer. Cont. Conf., pp. 201-206, June 1987.

[4] - R. Ortega, "On the Stability of Continuous-Time Adaptive Controllers for Pure Delay Systems," Systems and Control Letters, Vol. 8, pp. 23-27, 1986.

[5] - A.M. Hasanul Basher, R. Mukudan, and D.A. O'Connor, "Memoryless Feedback Control in Uncertain Dynamic Delay Systems," Int. J. Systems Sci., Vol. 17, No. 3, pp. 409-415, 1986.

[6] - A.S. Morse, "Ring Models for Delay Differential Systems," Automatica, Vol. 12, pp. 529-531, 1976.

SLIDING MODE PROPERTIES OF MIN-MAX CONTROLLERS AND OBSERVERS FOR UNCERTAIN SYSTEMS

STANISLAW H. ŻAK and RAYMOND A. DECARLO
School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

ABSTRACT

This paper reviews certain properties of min-max controllers used to stabilize uncertain dynamical systems and then develops parallels of these controllers with those of variable structure control. Conditions under which these min-max controllers behave as variable structure controllers are investigated. The paper generates sufficient conditions which guarantee global attractivity of the state trajectory to a naturally defined switching surface for a min-max control structure. Examples illustrating a phase plane interpretations of these conditions are included in the paper. Finally, applications of the obtained results to the design of a new type of state observers are discussed.

1. INTRODUCTION

Two recently developed methods of robust control and stabilization of dynamical systems in the presence of modelling errors and uncertain parameters are variable structure control (VSC) and the deterministic control of uncertain systems. VSC control uses a switched control strategy to drive the plant trajectory onto a prespecified so called switching surface in the state space and maintains the trajectory on this surface for all subsequent time. A trajectory, confined to such a switching surface, is said to be in a sliding mode in which case the system performance is insensitive to disturbances confined to a subspace of the state space. Determination of the control parameters which in fact force this behavior utilize the generalized Lyapunov second method. In the deterministic control of uncertain systems the plant uncertainties are represented by bounds on model parameters. The so called min-max controller structures ([1], [5]) which result in robust stabilization of these "uncertain" plants utilize the second method of Lyapunov for their specification. In addition these control strategies exhibit a switching nature with respect to a naturally defined surface dependent on the solution to a Lyapunov equation. Hence one expects a parallelism between the two control methodologies. This parallelism was also seen by Gutman and Palmor [5] and Chouinard et al. [1] where the attractivity to the naturally defined switching surface was considered. Chouinard et al. [1] presented a condition for maximization of a region of attractivity. This parallelism was also considered in [10].

After defining the model and reviewing the above referenced results, sufficient conditions for the attractivity of the naturally defined switching surface will be derived. Thus the min-max controller will behave as a VSC and the system restricted to the switching surface will exhibit a sliding mode behavior. These ideas will be illustrated by two examples. Finally, the paper examines applications of the obtained results to the design of a new class of min-max observers.

2. SYSTEM DESCRIPTION AND BACKGROUND RESULTS

Following the developments of [6] consider a state model

$$\dot{x}(t) = Ax(t) + B[u(t) + \xi(t,x(t))], \ \ x(t_0) = x_0 \tag{1}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $\xi(\cdot)$ represents the "lumped" uncertainties satisfying the norm condition

$$\|\xi(t,x(t))\| \leq \rho(t,x(t)) \triangleq \rho \tag{2}$$

where $\rho(\cdot): \mathbb{R}\times\mathbb{R}^n \rightarrow \mathbb{R}_+$ is a known nonnegative function, and $\|\cdot\|$ denotes standard Euclidean norm. The nominal system

$$\dot{x}(t) = Ax(t) \tag{3}$$

is assumed to be asymptotically stable in which case the matrix Lyapunov equation

$$A^TP + PA = -Q \tag{4}$$

has a solution, P, for any real symmetric positive definite matrix, Q, i.e. $Q = Q^T > 0$. Thus one can show that the closed-loop system is uniformly asymptotically stable in the large for any admissible uncertainty $\xi(\cdot)$ under the min-max control strategy

$$u = u(t,x) = \begin{cases} -\dfrac{B^TPx}{\|B^TPx\|}\,\rho(t,x) & \text{if} \quad \|B^TPx\| \neq 0 \\ \{\hat{u}\in R^m \mid \|\hat{u}\| \leq \rho(t,x)\} & \text{if} \quad \|B^TPx\| = 0\,. \end{cases} \tag{5}$$

It is important to note that the surface

$$S = \{x \mid \sigma(x) = B^TPx = 0\} \tag{6}$$

acts as a switching surface. It is our purpose to consider the attractivity of a state to this surface and develop sufficient conditions for global attractivity.

Definition [5]: A surface S is asymptotically attractive if (i) any solution starting on the surface remains there, and (ii) any solution starting outside the surface tends to it at least asymptotically.

If one defines the generalized Lyapunov function $V = \frac{1}{2}\sigma^T\sigma$, then a sufficient condition for attractivity to S is

$$\dot{V} = \frac{d}{dt}\left(\frac{1}{2}\|\sigma\|^2\right) < 0 \tag{7}$$

in a neighborhood of S [4]. Hence analogous to the control structure of (5) with due to reference to the surface S, one considers the structure

$$u = u(t,x) = \begin{cases} -\dfrac{\sigma}{\|\sigma\|}\,\bar{\rho} & \text{for} \quad \|\sigma\| \neq 0 \\ \{\hat{u}\in\mathbb{R}^m \mid \|\hat{u}\| \leq \bar{\rho}\} & \text{for} \quad \|\sigma\| = 0 \end{cases} \tag{8}$$

where $\bar{\rho}$ is to be determined. Hence when determining attractivity to the surface with control (8), it is sufficient that

$$\dot{V} = \sigma^T\dot{\sigma} \leq x^T[(PBB^TP)A + A^T(PBB^TP)]x - \lambda_{\min}(B^TPB)\|\sigma\|\bar{\rho} + \lambda_{\max}(B^TPB)\|\sigma\|\rho < 0\,. \tag{9}$$

Define $\hat{Q}$ as

$$\hat{Q} \triangleq -\left((PBB^TP)A + A^T(PBB^TP)\right)\,. \tag{10}$$

Hence from (9), S is globally asymptotically attractive if:

(i) $\hat{Q}$ is positive semidefinite and

$$\bar{\rho} > \frac{\lambda_{\max}(B^TPB)}{\lambda_{\min}(B^TPB)}\,\rho \tag{11}$$

or

(ii) $\hat{Q}$ is not positive semidefinite, but

$$\bar{\rho} > \frac{\lambda_{\max}(B^TPB)}{\lambda_{\min}(B^TPB)}\,\rho + \frac{\|B^TPAx\|}{\lambda_{\min}(B^TPB)}\,. \tag{12}$$

As such a VSC type behavior for the min-max control structure of (8) depends on the positive semidefiniteness of $\hat{Q}$. Thus if $\hat{Q}$ is positive semidefinite, then the controller requires less energy (as per equation (11) versus (12)) to achieve stability to the surface.

3. SUFFICIENCY CONDITIONS FOR THE POSITIVE SEMI-DEFINITENESS OF THE MATRIX $\hat{Q}$

The goal of this section is to show that a sufficient condition for $\hat{Q} \geq 0$ is that $N[B^TP]$ be A-invariant where $N[\bullet]$ means "null space of". We establish this condition by first showing that $\hat{Q} \geq 0$ if the nominal system is globally attractive to S (equation (6)) and then show that if $N[B^TP]$ is A-invariant, the nominal system is stable to S.

To determine the desired sufficiency conditions first observe that $\hat{Q} \geq 0$ if and only if $\sigma^T B^T P A x \leq 0$ where as before $\sigma = B^T P x$.

Consider now the nominal system

$$\dot{x} = Ax \ , \ \ x(0) = x_0 \ .$$

The switching surface $\sigma = B^T P x$ is asymptotically attractive for the nominal system if

$$\sigma^T \dot{\sigma} = \sigma^T B^T P \dot{x} = \sigma^T B^T P A x < 0 \quad \forall \ x \notin N[B^T P] \ ,$$

Thus the surface S is asymptotically attractive for the nominal system if $\hat{Q} \geq 0$.

Suppose now that it is possible to find a solution P to the matrix Lyapunov equation such that the rows of the matrix B^TP are the left eigenvectors of the matrix A corresponding to real eigenvalues $\lambda_i (i = 1,\ldots,m)$. Then we have

Proposition 1

If the rows of the matrix B^TP are the left eigenvectors of the matrix A corresponding to real eigenvalues λ_i $(i = 1,\ldots,m)$, then $\hat{Q} \geq 0$.

Proof: Let

$$B^T P \triangleq \begin{bmatrix} c_1^T \\ \vdots \\ c_m^T \end{bmatrix} \ , \ \ c_i^T \in \mathbb{R}^{1 \times n} \ .$$

where c_i, $i = 1,\ldots,m$, are appropriate left eigenvectors of A. By definition

$$c_i^T A = \lambda_i c_i^T \ , \ \ i = 1,\ldots,m \ .$$

Observe then that

$$\frac{1}{2} \, x^T \hat{Q} x = \sigma^T B^T P A x = \sigma^T \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix} B^T P x = \sigma^T \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix} \sigma$$

$$= \begin{bmatrix} \lambda_1 \sigma_1, \ldots, \lambda_m \sigma_m \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_m \end{bmatrix} = \sum_{i=1}^{m} \lambda_i \sigma_i^2 \leq \lambda_{i\max} \, ||\sigma||^2 \leq 0 \ ,$$

where $\lambda_{i\max} = \max\{\lambda_i \ , \ i = 1,\ldots,m\}$ and $\lambda_{i\max} = \lambda_{i\max}(A) < 0$ since the matrix A is asymptotically stable.

□

Consider now the following result of Chouinard et al. [1] which says:

Theorem 1 ([1] p. 388): There exists a positive definite P such that $N[B^TP] = N[B^TPA]$ if and only if there is a subspace W of dimension n-rank(B) satisfying $AW \subset W$ and $R(B) \cap W = \{0\}$ where $R(\cdot)$ means "range of".

The above theorem gives a necessary and sufficient condition for the existence of a solution of the matrix Lyapunov equation so that the switching surface $S = N(B^TP)$ is rotated in order to achieve a maximum region of attractivity to the surface.

Note that since $\det(B^TPB) \neq 0$ the columns of B are independent of $N(B^TP)$. Thus

$$R(B) \cap N(B^TP) = \{0\} .$$

Therefore we can take as W the subspace $N(B^TP)$ provided

$$A\, N(B^TP) = N(B^TP)$$

i.e. $N(B^TPA) = N(B^TP)$. Using the above results, it is now possible to prove the following

Proposition 2:

Let the row space of B^TP be spanned by a set of m left eigenvectors of A labelled $\{c_1,...,c_m\}$. Then $N[B^TPA] = N[B^TP]$, i.e. $N[B^TP]$ is A-invariant.

Proof: By the assumption, there exists a nonsingular matrix M such that

$$(MB^TP)A = \begin{bmatrix} c_1^T \\ \vdots \\ c_m^T \end{bmatrix} A = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix} \begin{bmatrix} c_1^T \\ \vdots \\ c_m^T \end{bmatrix} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix} MB^TP .$$

□

This proposition offers an additional characterization to that of Theorem 1.

We shall now demonstrate that if $N(B^TP)$ is A-invariant then

$$\sigma^TB^TPAx < 0 \quad \forall\, x \notin N[B^TP] .$$

Theorem 2: If $N(B^TP)$ is A-invariant then $S = \{x \mid \sigma(x) = B^TPx = 0\}$ is globally attractive for the nominal system.

Proof: Since $N[B^TP] \oplus R(B) = R^n$, $x = b + v$ where $b \in R(B)$ and $v \in N[B^TP]$. As such there exists α such that $b = B\alpha$. Since (B^TPB) is nonsingular there exists β such that $b = B(B^TPB)^{-1}\beta$. Multiplying on left by B^TP, a solution β to this equation is given by $\beta = B^TPb$. However since $v \in N[B^TP]$ we can choose $\beta = B^TP(b + v) = B^TPx$, i.e. $b = B(B^TPB)^{-1}B^TPx$. Therefore $x = b + v = B(B^TPB)^{-1}B^TPx + v$. This will be used below.

Using the generalized Lyapunov function $V = \frac{1}{2}\sigma^T(B^TPB)^{-1}\sigma$, observe that

$$\begin{aligned} \dot{V} &= \sigma^T(B^TPB)^{-1}B^TPAx = \sigma^T(B^TPB)^{-1}B^TPA[B(B^TPB)^{-1}\sigma + v] \\ &= [\sigma^T(B^TPB)^{-1}B^T](PA)[B(B^TPB)^{-1}\sigma] \text{ (by A−invariance)} \\ &= \frac{1}{2}\,[\sigma^T(B^TPB)^{-1}B^T](A^TP + PA)\,[B(B^TPB)^{-1}\sigma] \\ &= -\frac{1}{2}\,[\sigma^T(B^TPB)^{-1}B^T]Q[B(B^TPB)^{-1}\sigma] < 0 \,,\ \forall\, x \notin N[B^TP] \,, \end{aligned}$$

where Q is as defined in equation (4).

Corollary 1: If $N[B^TP]$ is A-invariant and $B^TPB = I_m$, then $\sigma^TB^TPAx < 0 \ \forall\ x \notin N(B^TP)$.

Observe that Theorem 2 and Corollary 1 determine sufficient conditions for attractivity to the surface in terms of the A-invariance of the null space of B^TP.

4. THE PHASE PLANE INTERPRETATION OF THE POSITIVE SEMIDEFINITENESS OF THE MATRIX $\hat{Q}$

In this Section we will confine our attention to second order systems with scalar controls modeled by the equation

$$\dot{x} = Ax + bu$$

where $A \in \mathbb{R}^{2\times2}$, $b \in \mathbb{R}^{2\times1}$. Since the dimension of $N(b^TP)$ is 1 for any nonsingular $P \in \mathbb{R}^{2\times2}$, the matrix A must to have real eigenvalues for the subspace $N(b^TP)$ to be invariant under A.

Example 1: Let

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

where $\lambda_i < 0$ $(i = 1,2)$. Define

$$c = b^TP = [\alpha,\beta]$$

where the matrix $P = P^T > 0$ is a solution of the matrix Lyapunov equation for some $Q = Q^T > 0$. Note that

$$-\hat{Q} = A^T(Pbb^TP) + (Pbb^TP)A = \begin{bmatrix} 2\lambda_1\alpha^2 & -(\lambda_1+\lambda_2)\alpha\beta \\ -(\lambda_1+\lambda_2)\alpha\beta & 2\alpha^2\beta^2 \end{bmatrix}.$$

The matrix $\hat{Q}$ is positive semidefinite iff all its principal minors are nonnegative. Hence the following inequalities must be satisfied

(i) $-2\lambda_1\alpha^2 \geq 0,$

(ii) $-2\lambda_2\beta^2 \geq 0$

(iii) $\det(\hat{Q}) = 4\lambda_1\lambda_2\alpha^2\beta^2 - (\lambda_1+\lambda_2)^2\alpha^2\beta^2 \geq 0.$

Since $\lambda_i < 0$ the inequalities (i) and (ii) are satisfied for any α and β. Inequality (iii) can be represented as follows

$$4\lambda_1\lambda_2\alpha^2\beta^2 - \lambda_1^2\alpha^2\beta^2 - 2\lambda_1\lambda_2\alpha^2\beta^2 - \lambda_2^2\alpha^2\beta^2 = -(\lambda_1-\lambda_2)^2\alpha^2\beta^2 \geq 0\ .$$

If $\alpha \neq 0$ and $\beta \neq 0$ then the above inequality will be satisfied iff $\lambda_1 = \lambda_2$.

The trajectories of the system

$$\dot{x} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} x\ ,\ \ \lambda_1 = \lambda_2\ ,$$

are the straight lines passing through the origin in the x_1,x_2 plane. Thus for any α, β the surface $\alpha x_1 + \beta x_2 = 0$ will be attractive everywhere.

Example 2: Let

$$A = \begin{bmatrix} 0 & 1 \\ -b^2 & k_2 \end{bmatrix}$$

where $k_2 < 0$, and let

$$c = b^TP = [1\ \ 1]$$

where, as in the previous example the matrix $P = P^T > 0$ is a solution of the matrix Lyapunov equation for some $Q = Q^T > 0$. The matrix $\hat{Q}$ has the form

$$\hat{Q} = -\left(A^T(Pbb^TP) + (Pbb^TP)A\right)$$

$$= \begin{bmatrix} 2b^2 & b^2 - 1 - k_2 \\ b^2 - 1 - k_2 & -2 - 2k_2 \end{bmatrix} .$$

$\hat{Q}$ is positive semidefinite if

(i) $2b^2 \geq 0$,

(ii) $k_2 \leq -1$, and

(iii) $-2b^2 - 2b^2k_2 \geq b^4 + 1 + 2k_2 + k_2^2$.

The last inequality can be represented as follows

$$[k_2 + (1 + b^2)]^2 \leq 0 .$$

Thus

$$k_2 = -\,(1 + b^2) ,$$

and

$$A = \begin{bmatrix} 0 & 1 \\ -b^2 & -(1 + b^2) \end{bmatrix} .$$

Note that the eigenvalues of A are $\lambda_1 = -1$, and $\lambda_2 = -b^2$, which means that we have a stable node at the origin with two asymptotes $\sigma_1 = x_2 - \lambda_1 x_1$ and $\sigma_2 = x_2 - \lambda_2 x_1$. Observe that the switching surface in our example coincides with the first asymptote.

If we now select

$$c = b^TP = [b^2 \;\; 1]$$

then after some manipulations we obtain $k_2 = -(1 + b^2)$, as before. This time our switching surface will coincide with the second asymptote. In both cases the switching surface is globally attractive whatever the initial position of the representative point in the phase plane.

5. OBSERVATION OF DYNAMICAL SYSTEMS IN THE PRESENCE OF BOUNDED UNCERTAINTIES AND SOME PROPERTIES OF MIN-MAX OBSERVERS

Consider a dynamical system as modeled by (1) and (2) with the output

$$y(t) = C\,x(t) , \tag{13}$$

where

$$C \in \mathbb{R}^{p \times n} , \;\; p \geq m .$$

Our goal is to design an observer with inputs y and u whose output $\bar{x}$ will converge to x, i.e.:

$$\lim_{t \to \infty} (x(t) - \bar{x}(t)) = 0 .$$

In what follows we present an observer which was proposed by Walcott and Żak [12].

The following assumptions will be utilized in the subsequent discussion:

A.1. The pair (A,C) is completely observable which implies that we can find a matrix $L \in \mathbb{R}^{n \times p}$ such that all eigenvalues of the matrix A-LC are open left half plane in the desired locations.

A.2. There exists a real symmetric positive definite matrix $Q \in \mathbb{R}^{n \times n}$ and a matrix $F \in \mathbb{R}^{m \times p}$ such that

$$FC = B^TP$$

where the matrix P is the unique, real symmetric positive definite solution of the Lyapunov matrix equation

$$(A - LC)^T P + P(A - LC) = -Q .$$

Let us now denote the error difference between the observer estimate and the true state by

$$e = \overline{x} - x . \tag{14}$$

Consider the following dynamical system

$$\dot{\overline{x}} = [A - LC]\,\overline{x} - M(e,\rho) + Ly + Bu , \tag{15}$$

where

$$M(e,\rho) = \begin{cases} \dfrac{B\,FCe}{\|FCe\|}\,\rho & \text{for all } e \notin S \\ \{m \in \mathbb{R}^n \mid \|m\| \leq \rho\} & \text{for all } e \in S , \end{cases} \tag{16}$$

and $S = \{e \mid \|FCe\| = 0\}$.

Theorem 3 [12]: Given the system modeled by (1), (2), and (13) and the observer governed by equation (15). If assumptions A.1. and A.2. are valid then

$$\lim_{t \to \infty} (\overline{x}(t) - x(t)) = \lim_{t \to \infty} e(t) = 0 .$$

Our goal now is to find conditions under which the trajectory of (15) is forced onto S and then is maintained on S for all subsequent time.

It turns out that the attractivity regions of S can be characterized in terms of the observer gain ρ. Specifically, consider the observer described by (15) where as in [13]

$$M = \begin{cases} \dfrac{B\,FCe}{\|FCe\|}\,\overline{\rho} & \text{for all } e \notin S \\ \{m \in \mathbb{R}^n \mid \|m\| \leq \overline{\rho}\} & \text{for all } e \in S \end{cases} \tag{17}$$

and $\overline{\rho}$ is to be determined. The equation describing the error difference between the output of the observer and the true state is

$$\dot{e} = A_0 e - \frac{B\,FCe}{\|FCe\|}\,\overline{\rho} - B\xi \tag{18}$$

for all $e \notin S$, where $A_0 \triangleq A - LC$. Note that (18) resembles (1) with the control strategy (8). Thus if we let $\sigma = FCe$ then performing manipulations analogous to those in Section 2 we arrive at the following conclusions. The surface $S = \{e \mid \|FCe\| = 0\}$ is globally asymptotically attractive if (i) $\hat{Q} = -\left((PBB^T P)A_0 + A_0^T (PBB^T P)\right)$ is positive semidefinite and

$$\overline{\rho} > \frac{\lambda_{\max}(B^T PB)}{\lambda_{\min}(B^T PB)}\,\rho$$

or (ii) $\hat{Q}$ is not positive semidefinite, but

$$\overline{\rho} > \frac{\lambda_{\max}(B^T PB)}{\lambda_{\min}(B^T PB)}\,\rho + \frac{\|B^T PA_0 e\|}{\lambda_{\min}(B^T PB)} .$$

Hence the behavior of the observer (15) depends on the positive semi-definiteness of $\hat{Q}$ and the gain ρ. Conditions under which the matrix $\hat{Q}$ is positive semidefinite can be deduced from Section 3. Thus the control problem and observer problem demonstrate their duality once again.

6. CONCLUSIONS

This paper presents sufficient conditions which guarantee global attractivity of the trajectories to "naturally" defined switching surface for min-max control and observer structures.

Examples illustrating a phase plane interpretation of some of these conditions are also included in the paper. The obtained results should be useful in the robust stabilization and state observation of uncertain dynamical systems. An interesting open problem is an extention of the results to a class of systems with nonlinear nominal part.

REFERENCES

[1] L.G. Chouinard, J.P. Dauer, and G. Leitmann, "Properties of matrices used in uncertain linear control systems," SIAM J. Control and Optimization, Vol. 23, No. 3, pp. 381-389, 1985.

[2] O.M.E. El-Ghezawi, A.S.I. Zinober, and S.A. Billings, "Analysis and design of variable structure systems using a geometric approach," Int. J. Control, Vol. 38, No. 3, pp. 657-671, 1983.

[3] B. Draženović, "The invariance conditions in variable structure systems," Automatica, Vol. 5, pp. 287-295, 1969.

[4] V.I. Utkin, "Variable structure systems with sliding modes," IEEE Trans. Automat. Contr., Vol. AC-22, No. 2, pp. 212-222, 1977.

[5] S. Gutman and Z. Palmor, "Properties of min-max controllers in uncertain dynamical systems," SIAM J. Control and Optimization, Vol. 20, No. 6, pp. 850-861, 1982.

[6] B.R. Barmish, M. Corless, and G. Leitmann, "A new class of stabilizing controllers for uncertain dynamical system," SIAM J. Control and Optimization, Vol. 21, No. 2, pp. 246-255, 1983.

[7] M.W. Spong and H. Sira-Ramirez, "Robust control design techniques for a class of nonlinear systems," Proc. 1986 American Control Conf., Seattle, WA, pp. 1515-1522, June 1986.

[8] S.M. Madani-Esfahani, R.A. DeCarlo, M.J. Corless, and S.H. Żak, "On deterministic control of uncertain nonlinear systems," Proc. 1986 American Control Conf., Seattle, WA, pp. 1523-1528, June 1986.

[9] I.R. Petersen, "Structural stabilization of uncertain systems: Necessity of the matching condition," SIAM J. Control and Optimization, Vol. 23, No. 2, pp. 286-296, 1985.

[10] R.A. DeCarlo, S.H. Żak, and G.P. Matthews, "Variable structure control of nonlinear multivariable systems: A tutorial," to appear: Proceedings of the IEEE.

[11] I.M. Gel'fand, "Lectures on Linear Algebra," Interscience Publishers, Inc., New York, 1961.

[12] B.L. Walcott and S.H. Żak, "State observation of nonlinear uncertain dynamical systems," IEEE Trans. Automat. Control, Vol. AC-32, No. 2, pp. 166-170, 1987.

[13] S.H. Żak, J.D. Brehove, and M.J. Corless, "Control of uncertain systems with unmodeled actuator and sensor dynamics and incomplete state information," submitted for publication.

ON THE STABILITY OF A MANIPULATOR INTERACTING WITH ITS ENVIRONMENT

JAMES E. COLGATE AND NEVILLE HOGAN
Massachusetts Institue of Technology
Cambridge, MA 02139

ABSTRACT

Dynamic interaction with the environment is fundamental to the process of manipulation. This paper motivates the need for "interaction controllers" and analyzes one aspect of their design, that of coupled stability. A necessary and sufficient condition for the stability of a linear manipulator coupled to a broad class of passive and active environments is found, and a simple test for coupled stability is presented. An analysis of the PID control of a second order system is provided to illustrate these concepts.

I. INTRODUCTION

The study of manipulation provides a strong incentive for understanding the control of interacting systems. Interaction, the exchange of mechanical work, is fundamental to the operation of controlled manipulators, such as the human arm or industrial robots. Despite the basic truth that this statement apparently offers, it has seldom been taken as an underlying principle for the design of robot controllers. On the contrary, most design methods emphasize trajectory following or static positioning, and most successful applications of industrial robots have been in areas such as spray-painting or welding for which the exchange of mechanical work is insignificant.

Yet, many industrial operations to which a robot might conceivably be applied, such as drilling, grinding, or cooperating with other robots in an assembly operation, require significant work transfer. A primary goal of the research described in this paper is to provide robotic manipulators with the functionality necessary to perform these sorts of tasks. Controllers designed to tailor the interactive behavior of a manipulator will be termed "interaction controllers."

The heirarchical structure of design goals shown in figure 1 may be used to provide a framework for the design of interaction controllers. As with conventional command-following controllers, nominal stability, i.e., the stability of the undisturbed closed-loop system, is paramount. But second only to this is coupled stability, i.e., the stability of the manipulator when it is interacting with an environment. A quantitative statement of the coupled stability specification will be the focus of this paper.

Nomenclature

a,b,c	PID controller parameters	u	control variable
A(s),B(s)	driving point impedances	v	velocity
B	viscous damping coefficient	x	position
f	force	Y(s)	admittance
F_e	force applied by environment	Z(s)	impedance
K	stiffness	ω	frequency, rad/sec
M	mass	τ	PID controller parameter
M_e	mass of environment		

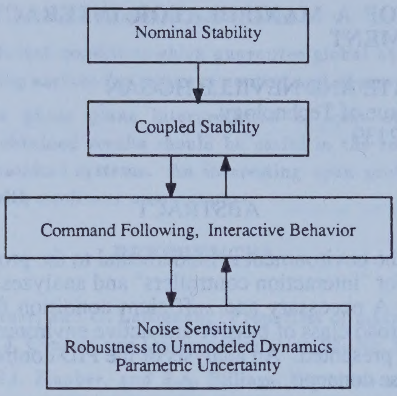


Figure 1. Design Specifications for an Interaction Controller

Beneath coupled stability are specifications for command following performance and interactive behavior. These will not be treated in this paper; however, approaches to specifying desired interactive behavior are presented in Hogan, 1985b,c, and Kazerooni, 1985. The final level of the heirarchy incorporates various robustness and sensitivity specifications; these will not be treated in this paper either, but are explored in Fasse, 1987. The arrows connecting the different levels of this heirarchy indicate that it is absolute at only the uppermost level. Tradeoffs between the other specifications will, in general, be required to optimize a given design.

Besides providing a framework for the design of interaction controllers, the heirarchy in figure 1 indicates that an approach to the design of interaction controllers differs from an approach to the design of command following controllers. In light of this difference, it should not be surprising that servo techniques, applied to the design of interaction controllers, do not always lead to the best results. This problem is illustrated in section III.

In the next section, a condition for the stability of a linear manipulator coupled to a passive environment [1] is derived. It is shown that this condition includes a wide class of active environments, and techniques for testing the coupled stability properties of a given design are presented.

II. COUPLED STABILITY

This section consists of an informal proof that a linear system must have a positive real driving point impedance[2] (the properties of a positive real function are described in the appendix) at any port where it couples to a passive environment if the coupled system is to be guaranteed stable. Certain interesting corollaries of this proof are also presented.

---

[1] The identification of the environment as a passive system rather than a system with admittance causality separates this work from "impedance control" in a strict sense; nevertheless, this work draws heavily upon the results of impedance control [Hogan, 1985a,b,c].

[2] A driving point impedance is a transfer function between port variables, such as force and velocity.

Consider two linear, time-invariant systems, each with an open interaction port:

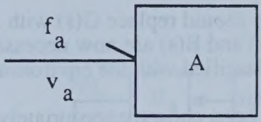


Figure 2

We are interested in the stability of the system that results when these two systems are coupled along their interaction ports, i.e., the stability of:

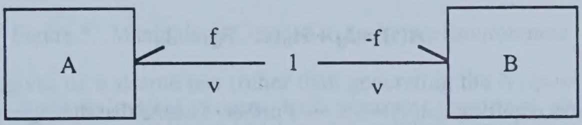


Figure 3

(Alternatively, these systems could be coupled with a 0 junction, in which case $f_a = f_b$ and $v_a = -v_b$; however, the stability result would be unaffected. See Rosenberg, 1983, for an explanation of 0 and 1 junctions).

Figure 3 may be rewritten in the following block diagram form:

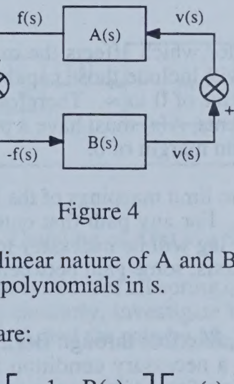


Figure 4

Implicit in this representation is the linear nature of A and B, i.e., both A(s) and B(s) may be expressed as the ratio of two rational polynomials in s.

The dynamics of the coupled system are:

$$\underline{0} = \begin{bmatrix} -1 & B(s) \\ -A(s) & -1 \end{bmatrix} \begin{bmatrix} v(s) \\ f(s) \end{bmatrix} \tag{1}$$

In order that a stable solution exist, the roots of the characteristic equation, $1 + A(s)B(s) = 0$, must fall within the closed LHP.

A Nyquist procedure may be used to study the pole locations. Given a plant G(s) and unity feedback, a statement of the Nyquist theorem is:

> The number of clockwise encirclements of the -1 point by a mapping through G(s) of the Nyquist contour (the Nyquist contour encloses the right half s-plane) equals the number of unstable closed loop poles (i.e., of 1+G(s)) within

the contour minus the number of unstable open loop poles (i.e., of G(s)) within the contour.

As we are interested in the roots of $1 + A(s)B(s)$, we should replace G(s) with A(s)B(s) in the analysis above. Some further assumptions about A(s) and B(s) are now necessary. First, A(s) is identified with our controlled system, and B(s) is identified with the environment.

Next, assume that the input and output of A(s) have been chosen appropriately so that a state space realization exists; i.e., so that F_a, G_a, H_a, and J_a exist which satisfy:

$$\begin{aligned} d\underline{x}/dt &= F_a\underline{x} + G_a v \\ f &= H_a\underline{x} + J_a v \end{aligned} \tag{2}$$

and:

$$A(s) = J_a + H_a(sI - F_a)^{-1}G_a \tag{3}$$

This is equivalent to requiring that $A(\infty) < \infty$. Further, assume that the eigenvalues of F_a lie in the closed left half plane. These assumptions, taken together, guarantee that the poles of A(s) lie in the closed left half plane, and are indicative of a stable system.

Finally, the environment is restricted to be linear and passive. As a consequence of representing the driving point impedance of a passive system, B(s) is a positive real function, and as such has the property that $\mathrm{Re}\{B(s)\} \geq 0$ for $\mathrm{Re}\{s\} \geq 0$ (see the appendix, condition 2). Thus, a mapping of the Nyquist contour through B(s) alone yields a curve which must lie wholly within the right half plane or on the imaginary axis. In fact, a mapping of the Nyquist contour through all possible positive real functions simply yields the entire closed right half B(s) plane.

If B(s) is viewed as a compensator which affects the magnitude and phase of A(s), then the class of all such compensators will include those capable of altering the phase by ±90°, and changing the magnitude by a factor of 0 to ∞. Therefore, it is clear that, in order to interact stably with all passive environments, A(s) must have a phase margin of ±90°, an upward gain margin of ∞, and a downward gain margin of 0.

The effect of these restrictions is to limit mappings of the Nyquist contour through A(s) to lie in the closed right half A(s) plane. For any path that enters the left half plane, no more than ninety degrees additional lead or lag will be necessary to cause the path to cross the negative real axis, and, once crossing the axis, some gain between zero and infinity will cause the path to encircle the -1 point.

Thus, the mapping through A(s), like that through B(s), is restricted to lie wholly within the closed right half plane. This is a necessary condition to ensure coupled stability, and is a necessary and sufficient condition for A(s) to represent the driving point impedance of a passive system [Brune, 1931]. Because the energy of this passive system may be selected as a Lyapunov function, it is clear, following the arguments of Fasse, 1987, that we also have a sufficient condition for stability when coupled to any passive environment, linear or nonlinear, that is describable in Hamiltonian (or Lagrangian) form. In conclusion, A(s) positive real is a necessary and sufficient condition to ensure stability when coupled to any passive, Hamiltonian environment.

This completes the proof, but there are certain related ideas which are useful. The first of these involves coupling to active linear environments. Figure 5 demonstrates how an active term may be added. It is evident that this active term has no effect on the stability of the loop unless u is in some way state-dependent. In fact, both A and B may be active without jeopardizing stable coupling if both satisfy the positive real condition when the state-independent control is turned off.

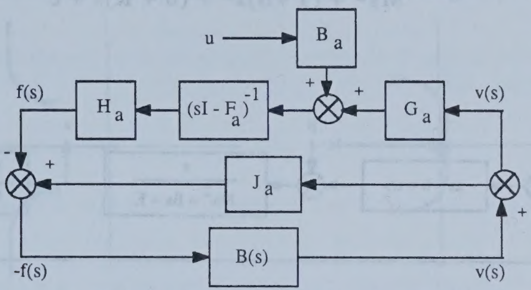


Figure 5. Manipulator Coupled to an Active Environment

The second idea gives us a simple test (other than generating the Nyquist plot of A(s)) for coupled stability. Because lossless systems, such as masses or springs, always exhibit ±90° phase, it is a lossless environment which is the most destabilizing passive environment for any A(s); any other passive environment will not rotate the Nyquist plot of A(s) as much. Further, because at any given frequency a lossless system exhibits the same behavior as an appropriately sized spring or mass, we need consider stability only when coupled to all springs and masses in order to test for stability when coupled to all passive environments.

This is a fairly simple test, given access to root locus software. Consider, for instance, the case in which A(s) is an admittance and the worst environment is some spring k_e or mass M_e. Coupling A(s) to the spring results in the following system:

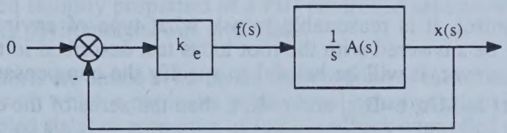


Figure 6. Coupling of A(s) to a Spring

We can investigate the stability of this system for all spring constants just by generating a root locus for the plant, A(s)/s. We can, similarly, investigate the stability of the system for all masses by replacing the plant with sA(s), and the gain by M_e.

III. AN EXAMPLE -- PID CONTROL OF A SECOND ORDER SYSTEM

Consider the PID control of a second order system described by the following state equations (M>0; B,K≥0):

$$\frac{d}{dt}\begin{bmatrix}x\\v\end{bmatrix} = \begin{bmatrix}0 & 1\\ -K/M & -B/M\end{bmatrix}\begin{bmatrix}x\\v\end{bmatrix} + \begin{bmatrix}0\\1/M\end{bmatrix}u + \begin{bmatrix}0\\1/M\end{bmatrix}F_e \qquad (4)$$

A block diagram representation of the closed loop system is shown in figure 7, and the driving point impedance (strictly speaking, admittance) of this system is:

$$Y(s) = \frac{s^2}{Ms^3 + (a + B)s^2 + (b + K)s + c} \qquad (5)$$

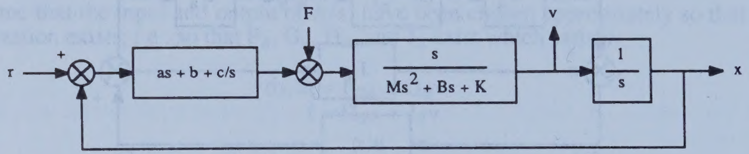


Figure 7. PID Control of a Second Order System

According to corollary 5 listed in the appendix, the zeros of a positive real function must be simple, a condition which is clearly violated by the double zero at the origin. A necessary condition for Y(s) to be positive real is c=0, in other words, the removal of all integral action. This is not very surprising in terms of conventional notions of passivity. The resulting admittance of the PD controlled system is:

$$Y(s) = \frac{s}{Ms^2 + (a + B)s + (b + K)} \qquad (6)$$

It is relatively easy to show that the PD controller results in a positive real Y(s) so long as $a>-B$ and $b>-K$. These are unrestrictive conditions, as it is nearly always the case that a and b are greater than zero (corresponding to negative feedback and a minimum phase compensator zero)

Returning to PID control, it is reasonable to ask what type of environment will lead to instability. This may be answered with the root locus test described in the previous section. Before doing this, however, it will be helpful to specify the compensator parameters more carefully. If we select $a=M/\tau$, $b=B/\tau$, and $c=K/\tau$, then the zeros of the compensator exactly cancel the poles of the plant, leaving the well behaved loop transfer function $1/(\tau s)$. τ is a free parameter which can be used to set the closed loop bandwidth. Given this choice of parameters, the driving point admittance is:

$$Y(s) = \frac{s^2}{(s + 1/\tau)(Ms^2 + Bs + K)} \qquad (7)$$

The coupled stability properties of this admittance can be examined by generating the root loci for spring environments and mass environments. These are shown for typical plant and compensator parameters in figure 8. Evidently, it is large masses that lead to difficulty. The minimum mass which will lead to instability may be found analytically to be:

$$(M_e)_{minimum} = 2B\sqrt{M/K} + B^2/K \qquad (8)$$

One conclusion that can be drawn from equation 8 is that high mass and damping and low stiffness lead to the most robust result, i.e., the largest environmental mass that will lead to instability. A stiff, underdamped system will be driven unstable by a small mass. This may also be seen on the root locus; the closer the complex poles move toward the imaginary axis, the smaller the mass that is needed to make the locus cross the imaginary axis.

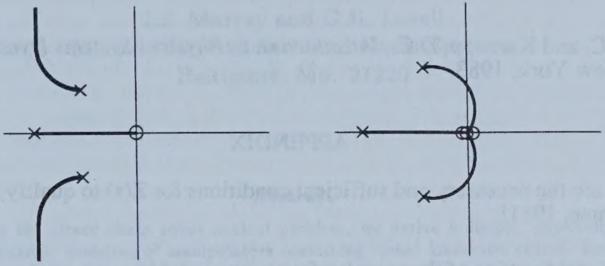


Figure 8. Controlled System Coupled to a Locus of Springs (left) and Masses (right)

Although this analysis is intended to serve primarily as an illustrative excercise, it may have significant implications for certain real problems. A space-based manipulator, for example, may be underdamped and may be expected, in the absence of gravity, to move large inertial loads. In this situation, integral control action would not be advisable.

IV. CONCLUSIONS

This paper has addressed one aspect of the problem of designing interaction controllers, that of coupled stability. A necessary and sufficient condition for the stability of a linear manipulator coupled to a passive, Hamiltonian environment was developed, and this condition was extended to include a broad class of active linear environments. A simple test for coupled stability, based on root locus, was also presented. Finally, these concepts were used to examine the coupled stability properties of a PID controlled second order system. A specific class of destabilizing environments was predicted.

Current research efforts are aimed at hardware interrogation of these concepts, the development of design methods with which a particular positive real behavior might be implemented, and an analysis of the coupled stability properties of force feedback controlled manipulators.

REFERENCES

Brune, O., "Synthesis of a Finite Two-Terminal Network whose Driving-Point Impedance is a Prescribed Function of Frequency," *J. Math. Phy., MIT*, **10**, August 1931, pp. 131-236.

Fasse, E.D., "Stability Robustness of Impedance Controlled Manipulators Coupled to Passive Environments," S.M. Thesis, Department of Mechanical Engineering, MIT, June, 1987.

Hogan, N., (1985a) "Impedance Control: An Approach to Manipulation: Part I -- Theory," *J. Dynamic Systems, Measurement, and Control*, **107**, March 1985, pp. 1-7.

Hogan, N., (1985b) "Impedance Control: An Approach to Manipulation: Part II -- Implementation," *J. Dynamic Systems, Measurement, and Control*, **107**, March 1985, pp. 8-16.

Hogan, N., (1985c) "Impedance Control: An Approach to Manipulation: Part III -- Applications," *J. Dynamic Systems, Measurement, and Control*, **107**, March 1985, pp. 17-24.

Kazerooni, H., "A Robust Design Method for Impedance Control of Constrained Dynamic Systems," Sc.D. Dissertation, Department of Mechanical Engineering, MIT, February, 1985.

Rosenberg, R.C. and Karnopp, D.C., *Introduction to Physical Systems Dynamics*," McGraw-Hill, New York, 1983.

APPENDIX

The following are the necessary and sufficient conditions for Z(s) to qualify as a driving point impedance [Brune, 1931]:

1. Z(s) is a rational function of the complex frequency variable $s = \sigma + j\omega$ and is analytic in general.
2. $\mathrm{Re}\{Z(s)\} \geq 0$ for $\mathrm{Re}\{s\} \geq 0$.
3. Z(s) is real when s is real.

Some of the corollaries of these conditions are:

1. Z(s) is a ratio of two polynomials in s with real coefficients.
2. The poles and zeros of Z(s) are restricted to the $j\omega$ axis and to the LHP.
3. Any poles which are not real must occur in complex conjugate pairs.
4. The poles on the $j\omega$ axis (including those at 0 and ∞) are simple with positive-real residues, i.e., for a pole at $j\omega_1$:
$$\lim_{s \to j\omega_1}[Z(s)(s - j\omega_1)] \geq 0$$
5. The zeros on the $j\omega$ axis (including those at 0 and ∞) are simple with positive differential quotients, i.e., for a zero at $j\omega_2$:
$$\lim_{s \to j\omega_2}[Z(s)/(s - j\omega_2)] \geq 0$$
6. The Nyquist plot of Z(s) must lie wholly within the closed RHP.

Closed Chain Manipulator Dynamics and Control

J.J. Murray and G.H. Lovell
Martin Marietta Aero and Naval Systems
Baltimore, Md. 21220

Abstract

To address the closed chain robot control problem, we derive a simple, physically insightful method for dynamic modeling of manipulators containing closed kinematic chains. Founded upon D'Alembert's principle, the method centers around a transformation from the well-understood dynamics of serial and open chain mechanisms to closed chain dynamic robot models. The framework for the closed chain dynamic robot model fosters the extension of serial robot engineering activities in the areas of digital simulation, real-time control, parameter identification, and optimal path planning to closed chain manipulators. We investigate the closed chain robot control problem and show that strategies for serial manipulator control are naturally extendible to closed chain manipulators.

1 Introduction

In contrast to open chain manipulators, closed chain manipulators have received little attention in the robotics literature. The increasing demand for legged vehicles, coordinated multi-manipulator systems, robotic hands, wheeled mobile robots, and faster, more precise manipulators motivates research in rigid body dynamics and control of closed chain robotic manipulators. Our goal in this paper is to draw upon the extensive knowledge of open chain manipulators and extend these dynamic modeling and control methodologies to closed chain manipulators. We present a simple, physically insightful transformation that relates open and closed chain dynamic robot models and establishes a framework for both analytical and algorithmic extensions of manipulator control methodologies.

A methodology for modeling the dynamic behavior of closed chain mechanisms that laid the foundation for our research was introduced by Wittenburg [32], through the notion of a *reduced system.* A reduced system is produced from a closed chain mechanism by hypothetically removing physical constraints until no closed kinematic chains remain. By definition, a reduced system is thus an open chain mechanism which can be routinely modeled using fundamental physical principles. The dynamic equations-of-motion for the closed chain mechanism are then derived from the reduced system dynamic model by replacing the removed constraints according to a standard Lagrange multiplier approach. Luh and Zheng [20] formalized this methodology for closed chain robotic manipulators to develop an *inverse dynamics* algorithm that computes the forces[1] at the actuated joints corresponding to a prescribed motion. A major contribution of their research is that the inverse dynamics computation for closed chain manipulators can be decomposed into two steps. First, the inverse dynamics of the reduced system is computed according to any serial formulation,[2] including the computationally efficient recursive Newton-Euler formulation [18]. Second, the generalized joint forces of the reduced system (computed in the preceeding step) are distributed among the actuated joints of the closed chain manipulator according to the method of Lagrange multipliers. Since the first step is more computationally intensive than the second step, substantial reductions in the first step significantly increase the overall efficiency of the closed chain inverse dynamics calculation. Thus, the numerical efficiency of the *serial* Newton-Euler formulation is exploited for the calculation of *closed chain* manipulator inverse dynamics.

Wittenburg [33] refined this methodology, applying the same principle of transforming reduced system dynamics into closed chain dynamics, but eliminating the need for Lagrange multipliers. A similar technique employed by Freeman [7] incorporates *intermediate* independent generalized coordinates, in which holonomic loop closure constraints are expressed naturally, to transform forces between systems under different constraints.

---

[1] We use the terms force and torque interchangeably.

[2] The application of serial dynamics formulations to model open chain structures is straightforward, albeit notationally cumbersome.

In this paper, we apply D'Alembert's principle [9] to formulate a transformation between the generalized actuating forces of a closed chain mechanism and the generalized actuating forces for a corresponding reduced system. The generalized actuating force transformation is mathematically equivalent to the aforementioned methodologies, but is developed independently of explicit dynamic equations-of-motion. This high level of abstraction yields a simple, physically insightful transformation that clarifies the relationship between closed chain mechanism and reduced system dynamics.

We apply this powerful analytic tool to demonstrate that dynamic modeling, simulation and control methodologies developed for serial manipulators are naturally extendible to closed chain manipulators. In fact, we show that the robot trajectory control problem for closed chain manipulators is essentially unchanged from the serial manipulator case apart from minor algorithmic details. This result also applies to *non-collocated* manipulators (i.e., manipulators for which positions and velocities can be measured at joints that are not actuated). We outline the algorithmic adjustments necessary to extend serial manipulator dynamic models, digital simulations, model-based and adaptive control laws, optimal path planners, and parameter estimation schemes to include closed chain manipulators.

This paper is organized as follows. In Section 2, we develop a generalized actuating force transformation between closed chain mechanisms and their respective reduced systems. We then apply this transformation in Section 3 to study dynamic models of manipulators containing closed kinematic chains. We assess the impact of closed kinematic chains on the robot trajectory control problem in Section 4. We then summarize our findings and contributions in Section 5, highlighting future applications and research opportunities in closed chain rigid body modeling and control.

2 Generalized Actuating Force Transformation

In this section, we formulate a transformation between

1. a set of generalized actuating forces for a set of holonomically constrained rigid bodies, and
2. a second set of generalized actuating forces for the same set of rigid bodies after some of the constraints are removed.

Our goal is to remove holonomic constraints until a dynamic model for the system of rigid bodies is easily derived, and then replace the removed holonomic constraints into this dynamic model through the application of the generalized actuating force transformation. Since this modeling procedure has potentially beneficial applications in all engineering disciplines that involve rigid body dynamics, we maintain a high level of generality in our development of the generalized actuating force transformation.

Consider a set of rigid bodies that is subject to workless constraints and is acted upon by applied forces and actuating forces. D'Alembert's principle permits inertial forces to be included as applied forces, and states that the necessary and sufficient condition for dynamic equilibrium is that no net virtual work be done by the applied and actuating forces for any virtual displacement consistent with the constraints. The virtual work done by the actuating forces must therefore balance the virtual work done by the applied forces. Consequently,

> *the virtual work done by the actuating forces for an arbitrarily chosen virtual displacement and a given set of applied forces is the same under any workless constraints that permit the virtual displacement.*

As the set of workless constraints is changed, the number of independent generalized coordinates required to represent the virtual displacement changes. The number, location, and orientation of the actuating forces may also change but must be physically realizable in the sense that D'Alembert's principle is satisfied.

To apply this form of D'Alembert's principle, we consider two mechanical systems, each with identical rigid bodies occupying identical positions in space and experiencing the same applied forces, but subjected to two different sets of workless constraints denoted by X and X_r. Let $\mathbf{q}$ and τ symbolize n-vectors of generalized coordinates and actuating forces for the system constrained by X. Similarly, $\mathbf{q}_r$ and τ_r denote m-vectors of generalized coordinates and actuating forces for the system constrained by X_r. Following the development of the preceeding paragraph, we observe that for any virtual displacement (consistent with both sets of constraints X and X_r) that is experienced by both systems, the virtual work done by the generalized actuating forces for both systems must be equal

$$\delta\mathbf{q}^T\tau = \delta\mathbf{q}_r^T\tau_r, \tag{1}$$

where $\delta\mathbf{q}$ and $\delta\mathbf{q}_r$ are two representations of the same virtual displacement for the set of rigid bodies.

We shall now apply the above result to a set of rigid bodies for which X comprises holonomic constraints and X_r is a proper subset of X. Further, let $\mathbf{q}$ and $\mathbf{q}_r$ be *independent* generalized coordinate vectors of the systems corresponding to the two sets of constraints. The respective systems possess n and m degrees-of-freedom (DOF), and we note that $n < m$. Now, if we assume that the set of rigid bodies only undergoes motions such that all the constraints X are satisfied, then $\mathbf{q}_r$ becomes functionally dependent on $\mathbf{q}$:

$$\mathbf{q}_r = \mathbf{f}(\mathbf{q}). \tag{2}$$

Physically, equation (2) represents the holonomic constraints that were removed from X to form X_r. In differential form, these constraints become

$$\dot{\mathbf{q}}_r = \mathbf{G}\dot{\mathbf{q}} \tag{3}$$

or

$$\delta\mathbf{q}_r = \mathbf{G}\delta\mathbf{q} \tag{4}$$

where the position dependent m x n *holonomic velocity constraint matrix* is

$$\mathbf{G} = \frac{\partial\mathbf{f}}{\partial\mathbf{q}}. \tag{5}$$

Substituting (4) into (1) yields

$$\delta\mathbf{q}^T\tau = \delta\mathbf{q}^T\mathbf{G}^T\tau_r. \tag{6}$$

Since the virtual displacement $\delta\mathbf{q}$ can be arbitrarily specified, it follows that

$$\tau = \mathbf{G}^T\tau_r. \tag{7}$$

The generalized forces corresponding to the (possibly different) actuating forces for the two systems are therefore related simply by a position dependent linear transformation. In the special case when the actuating forces for the two systems are identical, τ_r corresponds to a redundant set of actuating forces for the system constrained by X, and equation (7) simply becomes a classical transformation [2] between two sets of generalized actuating forces.

The generalized actuating force transformation in (7) is a key result that facilitates dynamic modeling of complex mechanical systems. Through the removal of holonomic constraints, the task of dynamic modeling can be greatly simplified. Once this dynamic model is derived, the effects of the removed constraints are reinstated according to the generalized actuating force transformation. Of particular interest in the robotics community is the application of this result to the dynamic modeling of manipulators containing closed kinematic chains.

3 Closed Chain Manipulator Dynamics

The generalized actuating force transformation developed in the preceeding section leads naturally to complete dynamic models of robotic manipulators containing closed kinematic chains. First, workless joints are virtually cut open to form an open chain reduced system. This procedure directly corresponds to removing holonomic constraints that "close the loops" in the manipulator. We note that there may be a number of ways to perform this task, each resulting in a unique open chain structure. After the dynamic model of the reduced system is derived, we apply the generalized actuating force transformation to reveal the explicit equations-of-motion for the closed chain manipulator. This modeling procedure provides a framework in which we detail how computationally efficient serial robot dynamics formulations are exploited for the numerical evaluation of the inverse and forward dynamics for closed chain manipulators.

A closed chain robotic manipulator consists of m rigid bodies (excluding a reference body) connected through single DOF rotational or translational joints, and possesses n DOF, where $n < m$. In practical applications, n of the single DOF joints are driven by the actuating forces τ, and the n-vector of corresponding joint displacements $\mathbf{q}$ is an independent set of generalized coordinates for the closed chain manipulator. The state of the manipulator is measured at another set of independent generalized cooordinates, symbolized by the n-vector $\mathbf{q}_m$. In the subsequent developments, the special case of collocated actuation and measurement is handled by simply substituting $\mathbf{q}$ for $\mathbf{q}_m$. Our starting point for the development of the dynamic model for a closed chain manipulator is a corresponding reduced system.

The open chain reduced system possesses m DOF and the m-vector of joint displacements $\mathbf{q}_r$ constitutes a set of independent generalized coordinates. Any standard open chain robot dynamics formulation [3,6,10,12,14,16,18,27] can be applied to model the reduced system using the generalized coordinates $\mathbf{q}_r$ and the corresponding generalized actuating forces τ_r. The dynamic model for the reduced system is a set of m coupled, nonlinear, second-order, ordinary differential equations:

$$\tau_r = \mathbf{D}_r(\mathbf{q}_r)\ddot{\mathbf{q}}_r + \mathbf{H}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r), \tag{8}$$

where $\mathbf{D}_r(\mathbf{q}_r)$ is the symmetric, positive definite m x m inertial coefficient matrix and $\mathbf{H}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r)$ is the m-vector representing centripetal, Coriolis, gravitational, and externally applied forces. We explicitly denote the functional dependences of $\mathbf{D}_r$ and $\mathbf{H}_r$ in (8), and throughout the remainder of this paper, to facilitate our developments.

The practice of virtually cutting workless joints open to form a reduced system is mathematically equivalent to removing holonomic loop closure constraints. Consequently, the constraints for the reduced system constitute a proper subset of the constraints for the closed chain manipulator. We can thus apply the generalized actuating force transformation that relates the closed chain dynamic robot model to the reduced system dynamic model in (8):

$$\tau = \mathbf{G}^T[\mathbf{D}_r(\mathbf{q}_r)\ddot{\mathbf{q}}_r + \mathbf{H}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r)]. \tag{9}$$

The above model completely characterizes the closed chain dynamics provided that the reduced system joint positions, velocities, and accelerations $\mathbf{q}_r$, $\dot{\mathbf{q}}_r$, and $\ddot{\mathbf{q}}_r$ are realizable by the closed chain mechanism. To make this restricition explicit in the model, we again apply the holonomic loop closure constraints represented in (2), using $\mathbf{q}_m$ as the independent generalized coordinates instead of $\mathbf{q}$:

$$\mathbf{q}_r = \mathbf{f}_m(\mathbf{q}_m). \tag{10}$$

The corresponding velocity and acceleration constraints are

$$\dot{\mathbf{q}}_r = \mathbf{G}_m\dot{\mathbf{q}}_m \tag{11}$$

and

$$\ddot{\mathbf{q}}_r = \mathbf{G}_m\ddot{\mathbf{q}}_m + \dot{\mathbf{G}}_m\dot{\mathbf{q}}_m. \tag{12}$$

The use of the measured joint coordinates $\mathbf{q}_m$ instead of the actuated joint coordinates $\mathbf{q}$ is physically motivated and accounts for the case when the measurements and actuation are not collocated. Applying these loop closure constraints to the dynamic model in (8), we obtain the complete dynamic model of the closed chain manipulator as a set of n coupled, nonlinear, second-order, ordinary differential equations:

$$\tau = \mathbf{G}^T[\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))(\mathbf{G}_m\ddot{\mathbf{q}}_m + \dot{\mathbf{G}}_m\dot{\mathbf{q}}_m) + \mathbf{H}_r(\mathbf{f}_m(\mathbf{q}_m), \mathbf{G}_m\dot{\mathbf{q}}_m)], \tag{13}$$

or, equivalently,

$$\tau = \mathbf{D}_m(\mathbf{q}_m)\ddot{\mathbf{q}}_m + \mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m), \tag{14}$$

where the n x n closed chain inertial coefficient matrix and the n-vector representing centripetal, Coriolis, gravitational, and externally applied forces are defined respectively by

$$\mathbf{D}_m(\mathbf{q}_m) = \mathbf{G}^T\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))\mathbf{G}_m \tag{15}$$

and

$$\mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m) = \mathbf{G}^T\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))\dot{\mathbf{G}}_m\dot{\mathbf{q}}_m + \mathbf{G}^T\mathbf{H}_r(\mathbf{f}_m(\mathbf{q}_m), \mathbf{G}_m\dot{\mathbf{q}}_m). \tag{16}$$

The closed chain dynamic robot model in (13) highlights the transformation of the reduced system dynamics into the dynamic equations-of-motion for a closed chain manipulator.

The holonomic loop closure constraints represented by (2) and the position dependent velocity loop closure constraint matrices $\mathbf{G}$ and $\mathbf{G}_m$ are central to the closed chain dynamic robot model in (13). The matrices $\mathbf{G}$, $\mathbf{G}_m$ and $\dot{\mathbf{G}}_m$ can always be systematically computed [32], although the holonomic loop closure constraints may only be expressible in implicit form. In many closed chain manipulators, these positional constraints can be derived in explicit form. When this is not possible, the joint displacements for the reduced system $\mathbf{q}_r$ can be computed from the measured joint displacements of the closed chain manipulator $\mathbf{q}_m$ through iterative techniques.

Table 1: Closed Chain Inverse Dynamics

1.	Compute $\mathbf{q}_r = \mathbf{f}_m(\mathbf{q}_m)$
2.	Compute $\dot{\mathbf{q}}_r = \mathbf{G}_m\dot{\mathbf{q}}_m$
3.	Compute $\ddot{\mathbf{q}}_r = \mathbf{G}_m\ddot{\mathbf{q}}_m + \dot{\mathbf{G}}_m\dot{\mathbf{q}}_m$
4.	Compute reduced system inverse dynamics according to an open chain formulation $\tau_r = \mathbf{D}_r(\mathbf{q}_r)\ddot{\mathbf{q}}_r + \mathbf{H}_r(\mathbf{q}_r, \dot{\mathbf{q}}_r)$
5.	Compute $\tau = \mathbf{G}^T\tau_r$

3.1 Closed Chain Inverse Dynamics

The inverse dynamics problem is to compute the actuating forces that correspond to a specified trajectory $\mathbf{q}_m$, $\dot{\mathbf{q}}_m$, and $\ddot{\mathbf{q}}_m$, according to the closed chain dynamic robot model in (13). Since the bracketed term on the right hand side of (13) is the inverse dynamics of the reduced system evaluated at $\mathbf{q}_r = \mathbf{f}_m(\mathbf{q}_m)$, $\dot{\mathbf{q}}_r = \mathbf{G}_m\dot{\mathbf{q}}_m$, and $\ddot{\mathbf{q}}_r = \mathbf{G}_m\ddot{\mathbf{q}}_m + \dot{\mathbf{G}}_m\dot{\mathbf{q}}_m$, we can immediately apply *open chain* algorithms (e.g., [14,16,18]). The actuating forces τ for the closed chain manipulator are then computed according to the generalized actuating force transformation in (7). The inverse dynamics calculation for closed chain manipulators is summarized, step-by-step, in Table 1.

3.2 Closed Chain Forward Dynamics

The forward dynamics problem is to compute the motion of a manipulator in response to specified actuating forces. In this section, we cast the closed chain forward dynamics problem into standard second-order explicit integration form, so that a wide variety of numerical integration techniques can be applied to determine the manipulator motion. In analogy with serial robot simulations [31], the independent generalized joint accelerations $\ddot{\mathbf{q}}_m$ are obtained by solving the n linear algebraic equations of the closed chain dynamic robot model:

$$[\mathbf{G}^T\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))\mathbf{G}_m]\ddot{\mathbf{q}}_m = \tau - \mathbf{G}^T[\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))\dot{\mathbf{G}}_m\dot{\mathbf{q}}_m + \mathbf{H}_r(\mathbf{f}_m(\mathbf{q}_m), \mathbf{G}_m\dot{\mathbf{q}}_m)], \tag{17}$$

or

$$\mathbf{D}_m(\mathbf{q}_m)\ddot{\mathbf{q}}_m = \tau - \mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m). \tag{18}$$

It can be shown that the closed chain inertial coefficient matrix $\mathbf{D}_m(\mathbf{q}_m)$ is invertible, and is symmetric, positive definite for collocated systems. These properties enable us to solve (17) for the generalized joint accelerations $\ddot{\mathbf{q}}_m$ through standard numerical techniques. Furthermore, we observe that the vector $\mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m)$ can be computed from the closed chain inverse dynamics algorithm in Table 1, setting the joint accelerations $\ddot{\mathbf{q}}_m$ equal to zero and using the current values of the measured joint positions $\mathbf{q}_m$ and velocities $\dot{\mathbf{q}}_m$. The inertial coefficient matrix $\mathbf{D}_r(\mathbf{f}_m(\mathbf{q}_m))$ of the reduced system can be computed according to standard *open chain* formulations (e.g., [31]), and leads naturally to the closed chain inertial coefficient matrix $\mathbf{D}_m(\mathbf{q}_m)$. The forward dynamics problem is thus cast into a standard second-order form that directly incorporates computationally efficient serial dynamics algorithms. We summarize the steps required to evaluate the joint accelerations $\ddot{\mathbf{q}}_m$ in Table 2. Since the above methodology incorporates a standard second order form and integrates only the independent generalized coordinates and velocities, numerical instabilities of integration schemes involving both dependent states and algebraic constraints [32] are obviated.

The closed chain dynamic robot model developed in this section permits the extension of serial manipulator simulation and control strategies to include closed chain manipulators. The inverse and forward dynamics for closed chain manipulators directly incorporate computationally efficient serial dynamics formulations, as detailed in Tables 1 and 2. Equipped with this understanding of the structure and salient properties of the closed chain dynamic robot model, we turn our attention to robot trajectory control and the impact of closed chains on established serial manipulator control methodologies.

Table 2: Closed Chain Forward Dynamics

1.	Compute $\mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m)$ from the closed chain inverse dynamics algorithm, setting $\ddot{\mathbf{q}}_m$ to zero
2.	Compute $\mathbf{b} = \tau - \mathbf{H}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m)$
3.	Compute $\mathbf{D}_r(\mathbf{q}_r)$ according to an open chain formulation
4.	Compute $\mathbf{D}_m(\mathbf{q}_m) = \mathbf{G}^T\mathbf{D}_r(\mathbf{q}_r)\mathbf{G}_m$
5.	Solve $\mathbf{D}_m(\mathbf{q}_m)\ddot{\mathbf{q}}_m = \mathbf{b}$

4 The Robot Control Problem Revisited

In this section, we consider the robot control problem for manipulators with closed kinematic chains. We show that parameter identification algorithms, optimal path planners, and model based and adaptive controllers designed for serial manipulators are naturally extendible to closed chain manipulators. The extensions for optimal path planners and model based controllers, including computed torque [21], resolved acceleration [19], sliding mode [2], suction [2], and operational space [13], are formulated in Section 4.1. The model based algorithms directly exploit the computational efficiency of our closed chain inverse dynamics algorithm. We address the closed chain extensions for adaptive controllers [4,11,15,29,30], including model reference and self-tuning approaches, and parameter estimators [1,24] in Section 4.2.

4.1 Model Based Control

From a model-based control viewpoint, the closed chain manipulator dynamics can be decoupled by nonlinear feedback according to the inverse dynamics. Our development of the closed chain dynamic robot model facilitates both design and implementation of model based control laws. Computed torque, resolved acceleration, sliding mode, suction, and operational space controllers are thus naturally extendible to closed chain manipulators. All of these controllers exhibit a common fundamental control law founded upon the inverse dynamics

$$\tau = \hat{\mathbf{D}}_m(\mathbf{q}_m)\mathbf{u} + \hat{\mathbf{H}}_m(\mathbf{q}_m, \dot{\mathbf{q}}_m), \tag{19}$$

where hats indicate the model resident in the controller, and $\mathbf{u}$ is dictated by the controller.

For example, the computed torque [21] controller for closed chain manipulators specifies $\mathbf{u}$ as

$$\mathbf{u} = \ddot{\mathbf{q}}_m^c + \mathbf{K}_v(\dot{\mathbf{q}}_m^c - \dot{\mathbf{q}}_m) + \mathbf{K}_p(\mathbf{q}_m^c - \mathbf{q}_m), \tag{20}$$

where $\mathbf{q}_m^c$ is the n-vector of commanded joint positions. If the controller model in (19) precisely matches the actual dynamics in (14), the system response is governed by

$$\mathbf{D}_m(\mathbf{q}_m)\mathbf{u} = \mathbf{D}_m(\mathbf{q}_m)\ddot{\mathbf{q}}_m. \tag{21}$$

Since the inertial coefficient matrix is nonsingular, the system is completely decoupled (when the gain matrices $\mathbf{K}_p$ and $\mathbf{K}_v$ are chosen to be diagonal) and responds as a linear system with the characteristic equation

$$\ddot{\mathbf{e}} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = \mathbf{0}, \tag{22}$$

where the joint error $\mathbf{e}$ is defined as the difference between the commanded and measured joint positions $\mathbf{e} = \mathbf{q}_m^c - \mathbf{q}_m$. The position and velocity gain matrices $\mathbf{K}_p$ and $\mathbf{K}_v$ are then selected such that the response meets performance specifications.

Resolved acceleration, sliding mode, suction, and operational space control are all characterized by the fundamental control law in (19), differing only in the specification of $\mathbf{u}$. In resolved acceleration control, the error is measured in Cartesian space rather than in joint space (as in the computed torque law in (20)). In sliding mode control, an extra term is appended to the computed torque law that causes the response of the system to remain on a sliding surface. Suction control operates in the same manner, save that the appended term causes the response of the system to remain inside a boundary layer around the sliding surface, eliminating chattering. Finally, operational space control is similar to resolved acceleration, but includes force feedback and control capabilities.

Paralleling real-time control of serial manipulators, the computational bottleneck of the aforementioned model based controllers is the inverse dynamics. Our inverse dynamics algorithm in Table 1 exploits the computational efficiency of recursive serial dynamic formulations and is immediately applicable to model based controllers for closed chain manipulators. The procedure consists simply of computing $\mathbf{u}$ in accordance with the control law to be applied (using the measured joint displacements $\mathbf{q}_m$ as the independent coordinates), and then substituting this $\mathbf{u}$ for $\ddot{\mathbf{q}}_m$ in step 3 of Table 1. A noteworthy observation is that the first two steps in Table 1 are unnecessary if all of the joint displacements $\mathbf{q}_m$, $\mathbf{q}_r$ and velocities $\dot{\mathbf{q}}_m$, $\dot{\mathbf{q}}_r$ are sensed and fed back to the controller.

4.2 Adaptive Control

Adaptive controllers appearing in the literature tend to be divided into three categories [11]: model reference, self-tuning, and linear perturbation control. Since the structure of the closed chain dynamic robot model in (14) is identical to that of the open chain model, the application of the majority of these adaptive controllers to closed chain manipulators is straightforward. The caveat is that adaptive control laws relying upon the symmetry and/or positive definiteness [11] of the inertial coefficient matrix may not be applicable to non-collocated closed chain manipulators.

A recent surge of adaptive control and parameter estimator research [1,4,24,29,30] exploits the fact that the serial manipulator actuating forces are linear in the constant dynamic parameters [22] (i.e., the link masses, center-of-mass vectors and inertia matrices) when these parameters are expressed in the appropriate coordinate frames. A significant property of the non-collocated closed chain dynamic robot model is that linearity in the constant dynamic parameters is retained. This property stems from the linearity of the reduced system actuating forces in (8) with respect to the constant dynamic parameters expressed in the Denavit-Hartenberg [5] link coordinate frames. Since a linear operation that is independent of the constant dynamic parameters is applied to transform the reduced system actuating forces into the non-collocated closed chain actuating forces, linearity in the constant dynamic parameters is retained. Thus, the considerable research efforts devoted to dynamic parameter estimation and adaptive control for serial manipulators are directly applicable to non-collocated closed chain manipulators. These controllers and estimators typically incorporate the inverse dynamics computation, which can be carried out for closed chain manipulators as detailed in Table 1.

5 Conclusions

While the past two decades has produced an abundance of engineering research into serial manipulator dynamics and control, the control of manipulators containing closed kinematic chains has not been adequately addressed. The structural and weight distribution advantages of closed chain manipulators lead naturally to lighter, faster robots [20]. To reap these benefits, we pay the price in the form of more complicated mechanical systems that must be accurately controlled. Our research concludes that extensions of serial manipulator dynamic modeling, simulation, and control for these more complicated closed chain systems are quite natural and inexpensive.

The analytical key to clarifying the issue of dynamic modeling and control for closed chain manipulators is the generalized actuating force transformation, which unveils a simple, physically insightful relationship between closed chain manipulator dynamics and the well-understood dynamics of open chain mechanisms. Dynamic models of closed chain manipulators are derived from the open chain reduced system equations-of-motion, and are thus easily interpreted for robot engineering activities such as digital simulation, real-time control, parameter identification, and optimal path planning. Our closed chain inverse and forward dynamics algorithms employ open chain dynamics formulations (which are incorporated in commercially available simulation software packages) as subalgorithms, as indicated in Tables 1 and 2.

A major contribution of this paper is the extension of trajectory control algorithms to include closed chain manipulators. The closed chain dynamic robot model provides a unified framework in which control laws for serial manipulators are analytically and algorithmically extended to include closed chain manipulators. We have shown how a wide spectrum of popular trajectory controllers can be extended to closed chain manipulators and identified possible pitfalls in extending certain controllers. A partial list of controllers that are naturally extendible to closed chain manipulators includes computed torque, resolved acceleration, sliding mode, suction, operational space, model reference, self-tuning, and linear perturbation control. The robot trajectory control problem (which includes digital simulation, real-time control, parameter identification, and optimal path planning) is thus essentially unchanged from the serial robot control problem, apart from minor algorithmic modifications.

The generalized actuating force transformation in Section 2 and the dynamic models in Section 3 are directly applicable to legged locomotion, coordinated multi-manipulator systems, and robotic hands. We anticipate that a major milestone in the dynamic modeling and control of these closed chain devices is the application of parallel computing architectures. The dynamics algorithms in Tables 1 and 2 immediately exploit the computational parallelism [14,16] of the open chain reduced system equations-of-motion. For closed chain systems, a second source of computational efficiency can be exploited: the virtually cut joints are selected to maximize the computational parallelism inherent in the reduced system. Clearly, a topic for future research is a systematic procedure for determining the joints to be virtually cut open. Other issues that are as yet unresolved are singularity identification and classification, the application of screw theory [6,17,28] to closed chain dynamics, and the systematic organization and customization [23,25,26] of closed chain dynamics algorithms to increase numerical efficiency.

6 Acknowldgements

The authors thank Mark Thomas, Abraham Orbach, Daniel W. Johnson and other associates who critiqued this manuscript for their helpful comments and suggestions.

References

[1] C. H. An, C. G. Atkeson, and J. M. Hollerbach, Estimation of Rigid Body Links of Manipulators, In *Proceedings of the 24-th IEEE Conference on Decision and Control*, Florida, December 11-13, 1985, pp. 990-995.

[2] H. Asada and J. E. Slotine, *Robot Analysis and Control*, John Wiley and Sons, New York, 1986.

[3] M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason (editors), *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.

[4] J. J. Craig, P. Hsu, and S. S. Sastry, Adaptive Control of Mechanical Manipulators, In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 7-10, 1986, pp.190-195.

[5] J. Denavit and R. S. Hartenberg, A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, *Journal of Applied Mechanics*, Volume 77(2), June 1955, pp. 215-221.

[6] R. Featherstone, The Calculation of Robot Dynamics Using Articulated-Body Inertias, *The International Journal of Robotics Research*, Volume 2(1), Spring 1983, pp. 13-30.

[7] R. A. Freeman, *Kinematic and Dynamic Modeling, Analysis and Control of Robotic Mechanisms (via Generalized Coordinate Transformation)*, Ph.D. Dissertation, University of Florida, 1985.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[9] D. T. Greenwood, *Principles of Dynamics*, Prentice-Hall, Englewood Cliffs, NJ, 1965.

[10] J. M. Hollerbach, A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamic Formulation Complexity, *IEEE Transactions on Systems, Man, and Cybernetics*, Volume SMC-10(11), November 1980, pp. 730-736.

[11] T. C. Hsia, Adaptive Control of Robot Manipulators - A Review, In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 7-10, 1986, pp. 183-189.

[12] T. R. Kane and D. A. Levinson, The Use of Kane's Dynamical Equations in Robotics, *The International Journal of Robotics Research*, Volume 2(3), Fall 1983, pp. 3-21.

[13] O. Khatib and J. Burdick, Motion and Force Control of Robot Manipulators, In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 7-10, 1986, pp. 1381-1386.

[14] R. H. Lathrop, Parallelism in Manipulator Dynamics, *The International Journal of Robotics Research*, Volume 4(2), Summer 1985, pp. 80-102.

[15] C. S. G. Lee and M. J. Chung, An Adaptive Control Strategy for Computer-Based Manipulators, In *Proceedings of 21st IEEE Conference on Decision and Control*, Orlando, FL, December 8-10, 1982, pp. 95-100.

[16] C. S. G. Lee and P. R. Chang, Efficient Parallel Algorithm for Robot Inverse Dynamics Calculation, *IEEE Transactions on Systems, Man, and Cybernetics*, Volume SMC-18(4), July/August 1986, pp. 532-542.

[17] G. H. Lovell, *A New Approach to Motor Calculus and Rigid Body Dynamics with Application to Serial Open-Loop Chains*, Ph.D. Dissertation, University of Florida, 1986.

[18] J. Y. S. Luh, M. W. Walker, and R. P. Paul, On Line Computational Scheme for Mechanical Manipulators, *Journal of Dynamic Systems, Measurement, and Control*, Volume 102(2), June 1980, pp. 69-76.

[19] J. Y. S. Luh, M. W. Walker, and R. P. Paul, Resolved-Acceleration Control of Mechanical Manipulators, *IEEE Transactions on Automatic Control*, Volume AC-35(3), June 1980, pp. 468-474.

[20] J. Y. S. Luh and Y. Zheng, Computation of Input Generalized Forces for Robots with Closed Kinematic Chain Mechanisms, *IEEE Journal of Robotics and Automation*, Volume RA-1(2), June 1985, pp. 95-103.

[21] B. R. Markiewicz, *Analysis of the Computed-Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator*, Technical Memorandum 33-601, Jet Propulsion Laboratory, Pasadena, CA, March 1973.

[22] J. J. Murray and C. P. Neuman, Computational Dynamic Robot Modeling, In *Proceedings of 27th Midwest Symposium on Circuits and Systems*, Morgantown, WV, June 11-12, 1984, pp. 479-481.

[23] J. J. Murray, *Computational Robot Dynamics*, Ph.D. Dissertation, Carnegie Mellon University, 1986.

[24] C. P. Neuman and P. K. Khosla, Identification of Robot Dynamics: An Application of Recursive Estimation, In *Proceedings of the Fourth Yale Workshop on Applications of Adaptive Systems Theory*, Yale University, New Haven, CT, May 29-31, 1985, pp. 42-49.

[25] C. P. Neuman and J. J. Murray, The Complete Dynamic Model and Customized Algorithms of the Puma Robot, *IEEE Transactions on Systems, Man, and Cybernetics*, Volume SMC-17(4), July 1987.

[26] C. P. Neuman and J. J. Murray, Customized Computational Robot Dynamics, *Journal of Robotic Systems*, Volume 4(4), August 1987.

[27] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Cambridge, MA, 1981.

[28] G. R. Pennock and A. T. Yang, Dynamic Analysis of a Multi-Rigid-Body Open-Chain System, *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Volume 105(3), March 1983, pp. 28-34.

[29] N. Sadegh and R. Horowitz, Stability Analysis of an Adaptive Controller for Robotic Manipulators, In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, March 31 - April 3, 1987, pp. 1223-1229.

[30] J. E. Slotine and W. Li, Adaptive Manipulator Control: A Case Study, In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, March 31-April 3, 1987, pp. 1392-1400.

[31] M. W. Walker and D. E. Orin, Efficient Dynamic Computer Simulation of Robotic Mechanisms, *ASME Journal of Dynamic Systems, Measurement, and Control*, Volume 104(3), September 1982, pp. 205-211.

[32] J. Wittenburg, *Dynamics of Systems of Rigid Bodies*, B. G. Teubner, Stuttgart, 1977.

[33] J. Wittenburg and U. Wolz, MESA VERDE: A Symbolic Program for Nonlinear Articulated-Rigid-Body Dynamics, ASME Paper No. 85-DET-151, 1985.

PLANNING A COLLISION - AVOIDANCE PATH USING ANALYTIC MODELING FOR OBSTACLES

Y.M. ENAB & J.Y.S. LUH
Department of Electrical and Computer Engineering,
Clemson University, Clemson, SC 29634-0915, U.S.A.

(1) INTRODUCTION

Customarily the industrial robots are defined as computer controlled mechanical manipulators used in industrial applications [1]. Robot tasks involve some manipulation requiring the travel of the end effector from point to another. An important problem in robotic manipulation among obstacles is that of planning collision-free trajectories. Luh [1] proposed a scheme using a "pillar" model of unexpected obstacles, constructed from the images from stereo cameras, and selecting a collision avoidance path from prestructured paths. Using a stereo cameras , however often involves convolution which is computationally time consuming. Recently, Luh [2] proposed a simple model, in which the physical obstacle is enclosed by a bounding box. When the Cartesian coordinates are defined to be parallel with the edges of the box , it can be described by the minimum and maximum values bounds of the obstacle on the three axes. In this paper a method for analytic modeling of three dimensional objects is introduced and a collision-avoidance path is designed.

(2) OBSTACLE MODELING

The new approach is to make use of the informations obtained using computer vision or any range finder method to determine the Cartesian coordinates for a set of points on the surface of the object. Assume these coordinates are $\{ x_i , y_i , z_i : i = 1,2,\dots,N \}$, where N is the number of points.

The primative used for object modeling is the ELLIPSOID because its equation is analytically known Also, the waste volume which will be inside the ellipsoid and does not belong to the physical obstacle, will be minimum. The complex objects can be modeled by more than one ellipsoid. The general equation of any point p(x,y,z) on the surface of the ellipsoid is

$$c_1 x^2 + c_2 y^2 + c_3 z^2 + c_4 xy + c_5 xz + c_6 yz + c_7 x + c_8 y + c_9 z = 1$$

where $\{ c_i \; ; \; i=1,2,..,9 \}$ are the equation coefficients . in matrix form , the above equation can be rewritten as

$$\underline{C}^T . \underline{X} = 1 \qquad \text{(T means transpose)} \qquad (1)$$

where $\underline{X}=[x^2 \; y^2 \; z^2 \; xy \; xz \; yz \; x \; y \; z]^T$ and $\underline{C}=[c_1 \; c_2 \; ...c_9]^T$ Using Least Mean Square estimation, we can prove that $\underline{C}^*$, the optimal solution for coefficient matrix is

$$\underline{C}^* = \underline{R}^{-1} \; \underline{U} \quad \text{where } \underline{R}= \sum_{i=1}^{N} \underline{X}_i \, \underline{X}_i^T \text{ and } \underline{U}= \sum_{i=1}^{N} \underline{X}_i \qquad (2)$$

Finally the ellipsoid equation is written as

$$\underline{C}^{*T} \underline{X}=G \quad \text{and} \quad G = 1 + MPE + W \qquad (3)$$

where MPE is the maximum positive error resulting in object modeling. This term is added to make sure that all the object points are on or inside ellipsoid surface. Also, W is robot arm width, this term enables us to model the links of robot arm by sticks and its end effector as a point. Now, it is easy to show that the inequality $\underline{C}^{*T} \underline{X} \leqslant G$ represents the forbidden region for the robot arm and the inequality $\underline{C}^{*T} \underline{X} > G$ represents the safe positions for it.

(3)COLLISION DETECTION

The collision between each link of robot arm and obstacle can be transformed to determining intersection between a straight line and ellipsoid. Assume the two ends of the straight line are the points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$, then the parametric equations of it are

$$x = a\lambda + b \quad , \quad y = h\lambda + d \quad , \quad z = e\lambda + f \qquad (4)$$

where: $a = x_2 - x_1$, $b = x_1$, $h = y_2 - y_1$, $d = y_1$, $e = z_2 - z_1$, $f = z_1$ and λ is the parameter. If $\lambda = 0$ then $p = p_1$ and if $\lambda = 1$ then $p = p_2$. Substituting by these equations into Eq. (1) we have

$$A\lambda^2 + B\lambda + E = 0 \qquad (5)$$

where: $A = c_1 a^2 + c_2 h^2 + c_3 e^2 + c_4 ah + c_5 ae + c_6 he$

$B = 2abc_1 + 2hdc_2 + 2efc_3 + c_4(ad+hb) + c_5(af+be) + c_6(hf+ed) + c_7 a + c_8 h + c_9 e$

$E = c_1 b^2 + c_2 d^2 + c_3 f^2 + c_4 bd + c_5 bf + c_6 df + c_7 b + c_8 d + c_9 f - G$

Intersection occurs only if $0 \leq \lambda \leq 1$.

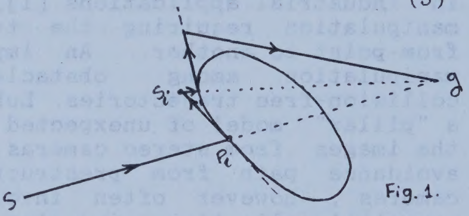


Fig. 1.

(4) COLLISION-AVOIDANCE PATH

The detailed algorithm is as follows.

step 1: Using the coordinates of starting point s and goal point g, determine the coefficients of the parametric equation of the straight line sg

Steb 2: Check the intersection between the ellipsoids and sg. If there is no intersection then the end effector can reach the goal point across sg safely. If there is intersection points choose the nearest one to s, which corresponds to the smallest λ. Assume this point p_ℓ.

Step 3: Determine the normal vector $\underline{n} = [u \ v \ 1]^T$ of ellipsoid surface at point p_ℓ using the following equations

$$u = -\partial z/\partial x = (2c_1 x + c_4 y + c_5 z + c_7)/(2c_3 z + c_5 x + c_6 y + c_9)$$

$$v = -\partial z/\partial y = (2c_2 y + c_4 x + c_6 z + c_8)/(2c_3 z + c_5 x + c_6 y + c_9)$$

The equation of the tangent plane at p_ℓ will be

$$(x - x_\ell)\,u + (y - y_\ell)\,v + (z - z_\ell) = 0$$

Step 4: Determine the coordinates of a point $s_\ell(x_\ell + \Delta x, y_\ell + \Delta y, z_\ell - u\Delta x - v\Delta y)$ on the tangent plane ($\Delta x, \Delta y$ are arbitrary)

Step 5: Consider s_ℓ as a new starting point and repeat steps 1 to 4 until the end effector reaches its goal g.

Through the above algorithm, the orientations of the end effector do not change until it reaches g. Also, feasibility of the path can be tested by check the intersection between the different links of the arm and the ellipsoids before its movement. Fig. 1. illustrates the algorithm in two dimensions.

(4)CONCLUSION

Ellipsoid is used as primative for modeling objects around robot arm. This simplifies collision detection and the analytic design of a collision-free path.

REFERENCES

1. Luh, J. Y. S., "A Scheme for Collision Avoidance with Minimum Distance Traveling for Industrial Robots," J. Robotic Sys., Vol., No. 1, pp. 5-26, Spring 1984.

2. Luh, J. Y. S. and Klaasen, J. A., "A Three-Dimensional Vision by Off-Shelf System with Multi-Cameras," IEEE Trans., Vol. PAMI-7, No. 1, pp. 35-45, January 1985.

STABILIZATION OF UNCERTAIN SYSTEMS SUBJECT TO HARD BOUNDS ON CONTROL WITH APPLICATION TO A ROBOT MANIPULATOR

MEHREZ HACHED, S. MEHDI MADANI-ESFAHANI, AND STANISLAW H. ŻAK
School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

ABSTRACT

A new class of controllers for linear time invariant uncertain systems is proposed. These controllers satisfy bound requirements on the control action. A transformation for decoupling the fast and slow states is utilized to investigate the stability estimates of the closed-loop system.

I. INTRODUCTION

In this paper we consider a class of linear time invariant multivariable control systems in the presence of uncertainties. A variable structure stabilizing controller with hard constraints on control is proposed. We utilize a transformation for decoupling the "fast" and "slow" states. Such a decomposition, simplifies the investigation of the stability of a system. We then find an estimate of the region where the Lyapunov-derivative of the "fast" subsystem is negative under two different assumptions: (i) the uncertainties in the system satisfy the so-called matching condition, (ii) the uncertainties are bounded by a positive constant. We then find an estimate of the region where the Lyapunov-derivative of the "slow" subsystem is negative under the assumption that the motion of the overall system when constrained to an m dimensional hyperplane is stable. The results found in the investigation of both the "slow" and "fast" subsystems are used then to obtain an estimate of the region of attraction of the overall system.

II. PROBLEM STATEMENT AND PRELIMINARY RESULTS

We consider systems modeled by differential equations of the form

$$\dot{x} = Ax(t) + Bu(t) + f(t,x), \ t \in [t_0,\infty) , \tag{1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $u \in \mathbb{R}^m$, and $f(\cdot,\cdot)$: $\mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a non linear vector function representing the perturbation in the system ($f(t,0) = 0, \forall t$). Concerning the system described by (1) we introduce the following assumptions.

Assumption 1: The function $f(\cdot)$ satisfies so-called matching condition; that is: there exists a function $g(\ , \ )$: $\mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that

$$f(t,x) = Bg(t,x) .$$

Assumption 2: The function $g(t,x)$ satisfies the condition

$$\|g(t,x)\| \leq L\|x\| , \ L > 0 , \ \forall \ t \in [t_0,\infty) .$$

Assumption 3: Matrix B is of full rank, i.e., rank B = m. Taking into account Assumption 1, the system (1) can be represented as follows

$$\dot{x}(t) = Ax(t) + B(u(t) + g(t,x)) . \tag{2}$$

Let $$y = Sx = \sigma(x) ,$$

where $S \in \mathbb{R}^{m \times n}$ and rank S = m. We also assume that $\det(SB) \neq 0$. In this paper we will use a linear transformation of the form

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} M_1 \\ S \end{bmatrix} x , \tag{3}$$

where $\det \begin{bmatrix} M_1 \\ S \end{bmatrix} \neq 0$ and $M_1B = 0$. Hence in the new coordinate the original system will have the form

$$\begin{bmatrix} \dot{z} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ SB \end{bmatrix} (u + \bar{g}) . \tag{4}$$

In most engineering problems, the designer is faced with hard bounds on the control action [1]. Let us assume that $|u_i| < \alpha_i$, $\alpha_i > i = 1,...,m$ and $K \triangleq \min[\alpha_i ; i = 1,...,m]$. We propose the following controller satisfying the above constraints.

$$u = \begin{cases} -K \dfrac{(SB)^{-1}\sigma(x)}{\|(SB)^{-1}\sigma(x)\|} & \sigma(x) \neq 0 \\ 0 & \sigma(x) = 0 . \end{cases} \tag{5}$$

Applying the control law (5) to the system (4) yields

$$\begin{aligned} \dot{z} &= D_{11}z + D_{12}y \\ \dot{y} &= D_{21}z + D_{22}y - \frac{Ky}{\|(SB)^{-1}y\|} + SB\,\bar{g}(t,z,y)\,. \end{aligned} \tag{6}$$

We will label the subsystem described by

$$\dot{z} = D_{11}z + D_{12}y \tag{7}$$

as the "slow" subsystem, and the subsystem described by

$$\dot{y} = D_{21}z + D_{22}y - \frac{Ky}{\|(SB)^{-1}y\|} + (SB)\bar{g}(t,z,y) \tag{8}$$

as the "fast" subsystem.

III. STABILITY ESTIMATES

Theorem 1 gives a sufficient condition for the stability of the system (6).

Theorem 1: The system (6) is stable in the region

$$\mathcal{R} = \{(z,y) : V(z,y) < \frac{1}{2}\, d_1\, \|y_0\|^2\,,\ y \in \mathbb{R}^m\,,\ z \in \mathbb{R}^{n-m}\}$$

where $$V(z,y) = \frac{1}{2}\, d_1\, y^T y + (1 - d_1) z^T P z\,,$$

$$\|y_0\| = \frac{k\,\alpha}{a(d + \alpha e)}\,,\quad d_1 = \frac{2}{2 + \alpha d}\,,$$

$$a = \|(SB)^{-1}\|\,,\quad d = \|D_{21}\| + L\|SB\|\,\|M^{-1}\|\,,$$

$$e = \|D_{22}\| + L\|SB\|\,\|M^{-1}\|\,,$$

$$\alpha = \frac{\min \lambda(Q)}{C \max \lambda(P)}\,,\quad C = \|D_{21}\|\,,$$

and $$PD_{11} + D_{11}^T P = -\,2Q\,.$$

Theorem 2: If the perturbation in system (1) satisfies the condition

$$|g_i| < \alpha_i\,,\ i = 1,\ldots,m\,,\ \|g(t,x)\| < \beta\,,\ 0 < \beta < \frac{K}{a\|SB\|}$$

then the system (6) is stable in the region

$$\mathcal{R}' = \{(z,y) : V'(z,y) < \frac{1}{2}\, d_1'\, \|y_0'\|^2,\ y \in \mathbb{R}^m,\ z \in \mathbb{R}^{n-m}\}\,,$$

where $$V'(z,y) = \frac{1}{2}\, d_1'\, y^T y + (1 - d_1') z^T P z\,,$$

$$\|y_0\|' = \frac{-\alpha\gamma}{d' + \alpha e'}\,,\quad d_1' = \frac{2}{2 + \alpha\, d'}\,,\quad d' = \|D_{21}\|\,,\quad e' = \|D_{22}\|\,,\quad \gamma = -\,\frac{K}{a} + \|SB\|\,\beta\,.$$

In the proof of the above theorems, we used [3], [4], and [5]. In the forthcoming paper, we made use of the theory developed in this paper to design the controller (5) for a two joint manipulator. We found an estimate of the region of attraction of the closed-loop system when the Coriolis terms are present in the system, and when they are ignored.

REFERENCES

[1] A. Weinreb and A.E. Bryson, "Optimal control of systems with hard control bounds," IEEE Trans. Automat. Contr., Vol. AC-30, No. 11, pp. 1135-1138, 1985.

[2] V.I. Utkin, "Sliding Mode and Their Application in Variable Structure Systems," Moscow, Mir, 1978.

[3] R.V. Patel, and M. Toda, "Quanlitative measures of robustness for multivariable systems, Proc. J.A.C.C., pp. TP8-A, San Francisco, CA, 1980.

[4] K.K.D. Young, P.V. Kokotović, and V.I. Utkin, "A singular perturbation analysis of high gain feedback systems," IEEE Trans. Automat. Contr., Vol. AC 22, No. 6, pp. 931-938, 1977.

[5] A. Saberi and H. Khalil, "Quadratic-type Lyapunov functions for singularly perturbed systems," IEEE Trans. Automat. Contr., Vol. AC-29, No. 6, pp. 542-550, 1984.

FINDING EFFECTIVE 'FORCE TARGETS' FOR TWO-DIMENSIONAL, MULTIFINGER FRICTIONAL GRIPS

JACOB T. SCHWARTZ(1) and M. SHARIR(1,2)

(1) Courant Institute of Mathematical Sciences,
New York University, New York, NY 10012
and
(2) School of Mathematical Sciences,
Tel Aviv University, Tel Aviv, ISRAEL

ABSTRACT: In this paper we consider the problem of calculating force targets for a collection of *n* fingers which grasp a two-dimensional object at known positions, at which the normals to the surface are also assumed to be known at least approximately. If the points at which the fingers touch the body do not allow a *positive* grip to be exerted (i.e. a grip in which the fingers hold the body in equilibrium by exerting friction-free forces in the directions of the corresponding inward-directed normals), it is appropriate to find the smallest coefficient of friction for which it is possible to assign a set of finger-force targets which hold the object at equilibrium and such that each individual force lies within the corresponding cone of friction. We present an algorithm for this problem which runs in time $O(n \log^3 n \log\log n)$. A modified version of this algorithm (involving preprocessing) is also given for the case in which the fingers need to balance a known external force and torque.

1. INTRODUCTION

The geometric problem analyzed in Sections 2 and 3 of this paper arises in the following context. We consider a robot 'hand' of some design, consisting of several independently movable, force sensing 'fingers'. For simplicity we shall suppose that these 'fingers' move as points in the plane and that each can sense the vector of force that it applies to any object which it contacts. (An experimental planar manipulator having these characteristics has been constructed in our laboratory.) The fingers are assumed to be quite stiff, but also assumed to have some small degree of compliance (relative to the positions to which they have been commanded to move); non-zero compliance is necessary if the fingers are to grip perfectly rigid bodies without infinite forces arising. Under these assumptions, what patterns of control should we apply to the motions of such gripping fingers, in order to cause them to move a gripped object along a specified path, while steadily applying a specified level of grip force?

This central pragmatic question defines the context addressed by this paper, and we begin by outlining the control approach that we have developed at our lab (which is similar in nature to approaches taken by several other researchers); see [Sc] for more details. It is assumed that the system to be controlled ('hand', plus the object which it is

---

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, by National Science Foundation Grant No. NSF-DCR-83-20085, by grants from the Digital Equipment Corporation, and the IBM Corporation. Work by the second author has also been supported by a research grant from the NCRD - the Israeli National Council for Research and Development.

gripping) is moving slowly enough for all inertial effects to be negligible. Then

A) The best available model of the system to be controlled is analyzed using the principles of elementary mechanics, leading to the definition of a 'good' set of gripping points and of a *force target* for each of the gripping fingers. This is the force which the finger is supposed to exert during the required motion (assuming that the external forces and torques on the object will remain constant during that motion).

B) Step (A) prepares for a *control loop*, which aims to keep the finger forces actually measured during an experiment close to their target values. If the discrepancy between model and actual system is not too large, this should cause the object O to follow a path close to that desired, while the forces exerted on various specified points of O remain near the target values specified in subphase (A).

The (finger-) force targets used in subphase (B) are derived by relating them either to various user specified *external force targets*, or to measurements of external forces (we assume that those measurements can be performed using the fingers themselves).

The fingers of the 'hand' are assumed to grip the contacted body frictionally. The force and torque acting on the body at any moment can then be calculated from the forces acting between the body and all of the gripping fingers; it is assumed that sensors for these latter forces are present in the fingers.

If we model the gripping fingers as points which can move in 2-space and make (frictional) contact with the gripped object, the only relevant aspects of the geometric situation are the positions at which the fingers are to contact the object, and the directions of the tangents to the object at these grasp points. Suppose first that the gripping points have already been chosen. Besides the evident equilibrium conditions (force/torque balance), to prevent slip we must insist that the force F_i exerted by the i-th gripping finger must lie in the cone of friction at the corresponding grip point x_i. Let μ be the coefficient of friction (which we assume to be the same for all fingers), and let $v_i^+(\mu)$ and $v_i^-(\mu)$ be the two boundaries of the friction cone at the i-th point, so that $v_i^+(\mu)=n_i+\mu t_i$ and $v_i^-(\mu)=n_i-\mu t_i$, where n_i (resp. t_i) is the unit normal (resp. unit tangent) to the gripped object at the i-th grip point. Then slip will be avoided as long as F_i belongs to the cone spanned by $v_i^+(\mu)$ and $v_i^-(\mu)$ for all i. The two equilibrium conditions that constrain F_i are

$$\sum F_i = 0 \quad \text{and} \quad \sum F_i \times x_i = 0 \, .$$

These conditions can be written as

$$\sum \alpha_i^+ p_i^+(\mu) + \sum \alpha_i^- p_i^-(\mu) = 0 \qquad (*)$$

where $p_i^+(\mu)$ (resp. $p_i^-(\mu)$) is the 3-dimensional vector $(v_i^+(\mu),(v_i^+(\mu)\times x_i)_3)$ (resp. $(v_i^-(\mu),(v_i^-(\mu)\times x_i)_3)$). We require a solution of the system (*) satisfying $\alpha_i^+\geq 0$, $\alpha_i^-\geq 0$ for all i (and of course not all the α_i's can be zero); hence the α_i's can be normalized to have a sum equal to 1, making it clear that what is needed is to express the zero vector as a convex combination of the vectors $p_i^+(\mu)$, $p_i^-(\mu)$.

Suppose that n fingers are involved. Then (inequalities aside) the $2n$ coefficients α_i^+, α_i^- are constrained only by the three linear conditions (*), which are far from determining them uniquely. To resolve this ambiguity in an advantageous manner, we propose to use the following heuristic. The nominal coefficient μ of friction must be regarded as somewhat uncertain, so it is appropriate to look for the smallest μ^* for which a positive solution α_i^+, α_i^- of the system (*) exists, and then to use

$$F_i = \alpha_i^+ v_i^+(\mu^*) + \alpha_i^- v_i^-(\mu^*)$$

as the target force for the i-th finger. (Note that the solution which best protects against friction coefficients smaller than assumed is also the solution which best protects against angular uncertainties in the tangent directions at the grip points.) When $\mu=0$, the friction cone degenerates to the vector n_i normal to the object surface at the point x_i, and our problem becomes that of expressing the zero vector as a convex combination of the vector $N_i=(n_i,(n_i \times x_i)_3)$.

In the preceding discussion we have assumed that the gripping positions x_i are fixed in advance. However, a more challenging variant of the problem is to find a good set of gripping positions, given some approximate representation of the object to be gripped, which can be acquired from visual or other sensory data. In the geometric problem that we study here, this issue will not be considered, and the reader is referred to a companion paper [MSS] for an initial study of this problem.

These observations define the problem studied in the following pages.

2. AN EFFICIENT GRIP FINDING ALGORITHM FOR A FREE OBJECT

The abstract problem to which the preceding discussion leads is as follows: Let $p_1(\mu), \ldots, p_n(\mu)$ be n points in 3-space, depending linearly on the real parameter $\mu \geq 0$. Let $K(\mu) = \text{conv}\,\{p_1(\mu), \ldots, p_n(\mu)\}$ be their convex hull, and let $C(\mu)$ be the cone generated by $K(\mu)$ with the origin as the apex (i.e. $C(\mu) = \{\lambda v : v \in K(\mu), \lambda > 0\}$). We assume that as μ increases so does $C(\mu)$. Our problem is to find the smallest μ^* for which the origin lies in $K(\mu^*)$. We present an algorithm for this problem which runs in time $O(n \log^3 n \log\log n)$.

We first simplify the problem. For each x,y and each μ define

$$M^+(x,y,\mu) = \min_i p_i(\mu)\cdot(x,y,1)$$

$$M^-(x,y,\mu) = \min_i p_i(\mu)\cdot(x,y,-1)$$

Note that there exists a point x,y for which either $M^+(x,y,\mu) > 0$ or $M^-(x,y,\mu) > 0$ if and only if the origin lies outside $K(\mu)$. Thus our task is to calculate the two quantities μ^+ and μ^- which are respectively defined as the smallest μ for which

$$\max_{x,y} M^+(x,y,\mu) \leq 0\ , \quad \text{respectively} \quad \max_{x,y} M^-(x,y,\mu) \leq 0\ .$$

The larger of these two μ's is the quantity μ^* that we seek.

Consider only the problem of finding μ^+, with symmetric handling of μ^-. For simplicity, write $M^+ = M$. Note that, for a fixed μ, calculating $\max_{x,y} M(x,y,\mu)$ is a linear programming problem in three dimensions, which can be easily accomplished in linear time, using Megiddo's elegant technique [Me1], [Me2], [Dy]. Moreover, if this maximum is positive (resp. nonpositive) we have $\mu^+>\mu$ (resp. $\mu^+\leq\mu$).

We combine this technique with the sophisticated parametric searching technique of Megiddo [Me3], which is briefly described below. We use this technique to find the value of μ^+ by simulating an execution of the linear programming algorithm on a generic, unknown value of μ^+. Every time when the algorithm needs to compare numbers which depend on μ^+, we determine the critical value(s) of μ at which the outcome of such a comparison can change, and then run the linear programming algorithm for these specific critical values to determine the position of the desired μ^+ with respect to these values, thus enabling us to determine the outcome of each of the comparisons in question, and to continue the execution of the generic algorithm. To save time, we try to use a parallel version of the generic algorithm which has a small depth $t(n)$ and a small number of processors $p(n)$. At each stage of the algorithm we will thus have at most $p(n)$ independent comparisons, giving rise to $O(p(n))$ critical values of μ. We can then perform binary search through all these critical values to locate μ^+ among them. The cost of the search is $O(n \log p(n) + p(n))$ -- the first term results from $O(\log p(n))$ binary search steps, each invoking the (serial) linear-time linear programming algorithm, and the second term is the search overhead. Repeating this for $t(n)$ stages, we get an overall performance of $O((n \log p(n) + p(n)) \cdot t(n))$. Each stage of this algorithm produces an interval which contains the desired μ^+, such that for all μ in that interval, each of the comparisons made at the current stage has a fixed outcome independent of μ. At the end of this simulated execution, we thus obtain a final interval I, namely the intersection of all these intervals, which must contain μ^+, and is such that for all $\mu \in I$, the execution of the generic algorithm proceeds through the same sequence of steps. It follows that $\mu^+ = \min I$.

More details about this elegant technique can be found in [Me3] and in the full version of the paper.

By examining the linear-time algorithm for linear programming in three dimensions, we see that it consists of $O(\log n)$ steps, each reducing the number of "surviving" planes, that are still relevant in finding $\max M(x,y,\mu)$, by a constant factor. Each step involves calculation of several medians (of orientations of certain intersection lines between the planes, and of the x and the y coordinates of certain sets of points of intersections between those lines), and a few invocations of a subprocedure for solving linear programming in two dimensions (for the given planes restricted over a certain line). This subprocedure also involves $O(\log n)$ steps, each calculating the median of certain intersection points between the restrictions of the given planes, and then calculating the minimum M and its derivative at this median.

We thus need to parallelize a generic version of this algorithm. As noted, it performs $O(\log^2 n)$ median-finding or minimum-finding calculations. Each such calculation can be done by a parallel algorithm that uses $O(n)$ processors and runs in $O(\log \log n)$ parallel time [AKSS], [Va] (see also [CY]). Plugging these procedures into the scheme described above, we obtain an overall time performance of $O(n \log^3 n \log \log n)$. Again this description of the algorithm is very condensed due to lack of space. More details can be found in the full version of the paper.

3. BALANCING AN ARBITRARY EXTERNAL FORCE AND TORQUE

The procedure of section 2 finds the 'optimal' frictional grip that can be exerted on a body not subject to outside forces. However, in many applications, gripping robot fingers will need to manipulate a body subject to varying external forces and torques (e.g. in the presence of gravity, or when the body makes contact with another object.) Assume that some sufficiently accurate measurement of these external forces and torques (F_e, τ_e) is available at each moment. A reasonable control heuristic is then to find the smallest coefficient of friction μ^* for which the 3-vector $\mathbf{e} = (F_e, \tau_e)$ can be expressed as a positive combination of the points $p_1(\mu^*), \ldots, p_n(\mu^*)$, i.e. the smallest μ^* for which $\mathbf{e}$ belongs to the cone $C(\mu^*)$. Since at this μ^* the vector $\mathbf{e}$ will lie on the boundary of $C(\mu^*)$, hence on a face of that cone generated by the origin and two points $p_i(\mu^*)$, $p_j(\mu^*)$, it follows that in general $\mathbf{e}$ will be expressible as a positive combination of just two of the vectors p_i at the optimal μ^*.

For each $\mu < \mu^*$ the vector $\mathbf{e}$ lies outside the cone $C(\mu)$. It is easy to check that this is equivalent to the existence of a plane π separating $\mathbf{e}$ from $C(\mu)$ and supporting $C(\mu)$ at the origin. Let $\mathbf{x}$ be the vector normal to π and pointing into the side of π containing $C(\mu)$. Then we must have

$$\mathbf{e} \cdot \mathbf{x} \le 0$$

$$p_i(\mu)\cdot \mathbf{x} \geq 0\ , \qquad i = 1, \ldots, n\ .$$

Two approaches to the problem suggest themselves. One is to calculate

$$\max_{\mathbf{x}} \min_{i=1}^{n+1} p_i(\mu)\cdot \mathbf{x}\ ,$$

where we have put $p_{n+1}(\mu) \equiv -\mathbf{e}$, and then find the smallest value of μ for which this maximum is nonpositive. This can be done in exactly the same way as in the preceding section, at computational cost $O(n \log n \log\log n)$. However, this solution requires application of this procedure each time the external force-and-torque vector $\mathbf{e}$ changes, i.e. once each control cycle, and is therefore prohibitive. Alternative approaches are therefore desirable. These can involve (possibly somewhat costly) preprocessing before object manipulation begins, and then use the structures generated by preprocessing to find the optimal μ and the balancing finger-force targets quickly during each control cycle.

Again, for lack of space, we provide only a brief sketch of our solution. We can reduce the problem to a 2-dimensional one, by taking a cross-section of $C(\mu)$ through some appropriate fixed plane P, and projecting (perspectively from the origin) each $p_i(\mu)$ and the "query" force/torque vector $\mathbf{e}$ onto P. The problem then becomes: Given n points $q_1(\mu), \ldots, q_n(\mu)$ in the plane, which depend as linear rational functions on μ, preprocess them so that for each query vector x, we can find quickly the smallest μ for which x belongs to the convex hull $K(\mu)$ of the points $q_i(\mu)$.

We provide a careful analysis of the pattern in which $K(\mu)$ changes combinatorially as μ increases. This happens at critical values of μ in which three vertices of $K(\mu)$ become collinear. A naive bound on the number of such critical events is $O(n^3)$, but we show that the actual number if only $O(n\,\lambda_5(n))$, where $\lambda_s(n)$ is the maximum number of maximal connected graph portions of n continuous functions, which can appear in their lower envelope, where it is assumed that each pair of these functions intersect in at most s points. It was shown in [HS], [ASS] (see also [Sh], [Sh2]) that $\lambda_s(n)$ is a nearly linear function of n for any fixed s. For $s=5$ it follows from [ASS] that $\lambda_5(n) = O(n\alpha(n)^{O(\alpha(n))})$, where $\alpha(n)$ is the very slowly growing inverse of the Ackermann function. (A similar bound was also obtained by Atallah [At], using a different technique.)

We also show how to calculate all these critical values of μ, and construct from them an appropriate data structure which allows us to answer "force target" queries as above efficiently. Specifically, we show

Theorem: The $O(n)$ finger force/torque vectors $p_i(\mu)$ can be preprocessed in time $O(n\lambda_5(n)\log n)$ to produce a data structure of size $O(n\lambda_5(n))$, from which, in $O(\log n)$ time, we can determine the smallest coefficient of friction μ for which any given external force/torque vector $\mathbf{e}$ is expressible as a positive combination of (two of) the vectors $p_i(\mu)$, and also produce such a positive combination when it exists.

References

[ASS] P. Agarwal, M. Sharir and P. Shor, Improved bounds for the length of general Davenport Schinzel sequences, in preparation.

[AKSS] M. Ajtai, J. Komlos, W.L. Steiger and E. Szemeredi, Deterministic selection in $O(\log \log n)$ parallel time, *Proc. 18th ACM Symp. on Theory of Computing,* 1986, pp. 188-195.

[At] M. Atallah, Dynamic computational geometry, *Proc. 24th Symp. on Foundations of Computer Science,* 1983, pp. 92-99. Also in *Comp. Math. with Appls.* 11 (1985) pp. 1171-1181.

[CY] R. Cole and C.K. Yap, A parallel median algorithm, *Inf. Proc. Letters* 20 (1985) pp. 137-139.

[Dy] M.E. Dyer, Linear time algorithms for two and three variable linear programs, *SIAM J. Computing* 13 (1984), pp. 31-45.

[HS] S. Hart and M. Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (1986) pp. 151-177.

[Me1] N. Megiddo, Linear time algorithms for linear programming in $\mathbf{R}^3$ and related problems, *SIAM J. Computing* 12 (1983) pp. 759-776.

[Me2] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* 31 (1984) pp. 114-127.

[Me3] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. Assoc. Comput. Mach.* 30 (1983) pp. 852-866.

[Sc] J.T. Schwartz, Comments on control primitives for manipulation, manuscript, December 1985.

[Sh] M. Sharir, Almost linear upper bounds on the length of general Davenport Schinzel sequences, *Combinatorica* 7 (1987) pp. 131-143.

[Sh2] M. Sharir, Improved lower bounds on the length of Davenport-Schinzel sequences, to appear in *Combinatorica.*

[Va] L.G. Valiant, Parallelism in comparison problems, *SIAM J. Computing* 4 (1975) pp. 348-355.

A Polygonal Approach to Hidden-Line Elimination

Michael T. Goodrich
Dept. of Computer Science, The Johns Hopkins University, Baltimore, MD 21218

Summary of Results

We give an algorithm for the well-known hidden-line elimination problem from computer graphics. Our algorithm runs in $O(n \log n + k + t)$ time, where n is the number of edges, and k (resp. t) is the number of intersecting pairs of edges (resp. polygons) in the projection plane π. Our algorithm is optimal in the worst case, and is also able to take advantage of problem instances which are "simpler" than in the worst case. If the the polygons in π are rectilinear, then we can solve the problem in $O(n \log n \log\log n + k)$ time. Both of our algorithms use $O(n+k)$ space, and only make one global visibility test per polygon.

Keywords: Computational geometry, computer graphics, hidden-line elimination.

1 Introduction

The hidden-line elimination problem [5,10,12,13,14,17,15,16,18] is well known in computer graphics: given a set of simple, non-intersecting planar polygons in 3-dimensional space, and a projection plane π, we wish to determine which portions of the polygonal boundaries are visible when viewed in a direction normal to π. Using the terminology of [18], we are interested in the *object space* version of the problem. That is, we want a solution which solves the problem independent of a specific rendering device.

We briefly review some of the efficient algorithms for this problem. In [5] Dévai gives an algorithm running in $O(n^2)$ time and $O(n^2)$ space, as does McKenna in [12] (McKenna's algorithm is actually for hidden-surface elimination, but it can be used for hidden-lines as well). Both of these algorithms are optimal in the worst case, because there are problem instances which have $\Omega(n^2)$ output size [5,12]. However, these algorithms take $O(n^2)$ time even if the size of the output is small (e.g., $O(1)$). In [13] Nurmi gives an algorithm which runs in $O((n+k)\log n)$ time and $O((n+k)\log n)$ space, where k is the number of intersecting pairs of edges in π (k is at most $O(n^2)$). Schmitt [15] is able to achieve this same bound using only $O(n+k)$ space. When the number of intersecting edges is not too large (i.e., $k << n^2/\log n$), these algorithms clearly run faster than $O(n^2)$. They are not worst-case optimal, however. In [10] Güting and Ottmann address special cases of the hidden-line elimination problem, namely, when the polygons are *c-oriented* in $\Re^3$, i.e., each edge is in one of a constant number of orientations in $\Re^3$, and, in particular, when they are 2-oriented. They give algorithms which run in $O(n \log^2 n + k)$ time, which, for these special cases, are optimal in the worst case and also take advantage of problem instances which are "simpler" than in the worst case.

Recently, Chazelle and Edelsbrunner [4] have shown how to construct the graph of segment intersections in π in optimal $O(n \log n + k)$ time. Unfortunately, this does not lead to improvements in any of the algorithms listed above. One exception to this is an algorithm by Schmitt [16] which runs in $O(n + k + r)$ time given the intersection graph, where r is the number of $(edge, polygon)$ intersections in π. His algorithm makes a global visibility test for each edge, however, making his algorithm less desirable from a practical point of view.

In this paper we give an algorithm for the general hidden-line elimination problem which is optimal in the worst case, and also takes advantage of problem instances which are "simpler" than in the worst case. Intuitively, our approach is to exploit the polygonal properties of the input, whereas previous algorithms concentrate more on edges. Specifically, our algorithm for the general hidden-line elimination problem runs in $O(n \log n + k + t)$ time, where t is the number of $(polygon, polygon)$ intersections (which is at most $O(n^2)$). Note that t is always less than r, the number of (*edge,polygon*) pairs. In fact, r can be significantly larger than t in general, e.g., one can easily construct examples where t is $O(n)$ while r is $\Omega(n^{3/2})$. We also show how to efficiently solve the hidden-line elimination problem for the case when the polygons in the projection plane are 2-oriented. Polygons of this type could arise, for example, from VLSI problems. Our algorithm for this case runs in $O(n \log n \log\log n + k)$ time. Both of our algorithms use $O(n + k)$ space.

One of the key ideas present in both of our algorithms is a method which allows us to perform only one global visibility test for each polygon, whereas previous methods essentially performed a global visibility test for each polygonal vertex and each intersection of two polygonal boundaries. Besides being computationally efficient, our algorithms are also conceptually simple, making them of both theoretical and practical worth.

This paper is divided into three sections. In the next section we describe the general hidden-line elimination algorithm and in Section 3 we show how to solve the hidden-line elimination problem for the case when the input polygons are 2-oriented in the projection plane.

2 The General Algorithm

We begin by defining the P-arrangement of a set of polygons, a structure which is a subgraph of the graph of segment intersections as constructed by Chazelle and Edelsbrunner [4] with extra information stored at the nodes and edges. Let a set $S_\pi = \{P_1, P_2, \ldots, P_n\}$ of simple polygons in the plane π be given. For any polygon P_i we let ∂P_i denote the boundary of the polygon P_i, and we define the *representative vertex* of P_i, denoted $rep(P_i)$, to be the vertex with smallest y-coordinate from all the vertices of P_i with smallest x-coordinate, i.e. $rep(P_i)$ is the vertex which would first if the vertices of P_i were sorted in increasing order lexicographically by (x, y)-coordinates. For any point p, we define the *vertical shadow* of p in S_π to be the first point which is intersected by the vertical ray emanating downward from p and which belongs to the boundary of a polygon in S_π. We define the *P-arrangement* of S_π to be the embedded planar graph $G = (V, E)$, where we have the following:

1. V consists of all nodes v which satisfy one of the following:
   (a) v is a vertex of a polygon in S_π,
   (b) v is an intersection point of the boundaries of two polygons in S_π, or
   (c) v is the vertical shadow of the representative vertex of a polygon in S_π;

2. E consists of all the (undirected) pairs (v, w), $v, w \in V$, which satisfy one of the following:
   (a) v and w are connected by a polygonal edge s and there is no point $z \in V$ between v and w on s, or
   (b) w is the vertical shadow of v and v is the representative vertex of some polygon in S_π.

In addition, we store the all the representative vertices which do not have vertical shadows in S_π in a set R. With each vertex v in V we store the names of all the polygons P such that v belongs to ∂P. For each edge $(v, w) \in E$ we mark (v, w) as to whether it belongs to the boundary of a polyon in S_π or not, and if so, we store which side of (v, w) is in the interior of that polygon. For simplicity of expression, we assume that each edge is adjacent to at most two polygons, one on each side. One can allow for more general configurations by allowing for different edges to overlap. See Figure 1 for an example P-arrangement.

In the following lemma we study the time and space complexity of constructing the P-arrangement of a set of polygons.

Lemma 2.1 *Given a set S_π of simple polygons in the plane π, the P-arrangement for S_π can be constructed in $O(n \log n + k)$ time and $O(n + k)$ space, where n dentoes the number of polygonal edge segments in S_π and k is the number of pairs of intersecting segments in π.*

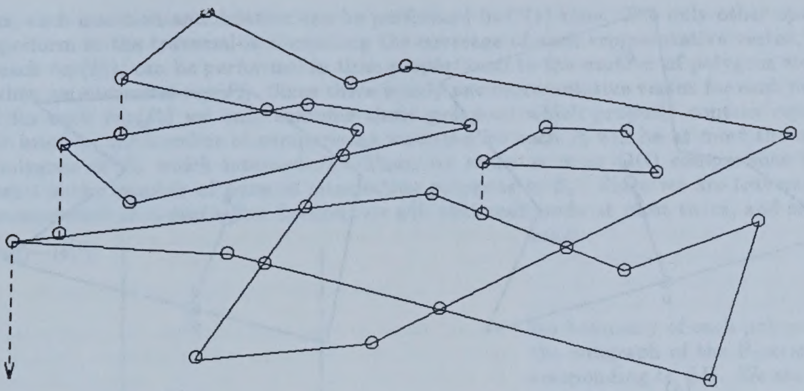


Figure 1: An example P-arrangement.

Proof: One first constructs the graph of segment intersections in $O(n \log n + k)$ time and $O(n+k)$ space using the method of Chazelle and Edelsbrunner [4]. Their method also gives the vertical shadows for each vertex. Thus, we can construct the graph G and the set R in $O(n+k)$ time given this segment intersection graph. It is a simple matter to then associate the extra information, as outlined above, with each node and edge in G in $O(n+k)$ additional time. ∎

We next describe how we use the P-arrangement to do hidden-line elimination.

2.1 A High-Level Description

We are given a set S of simple, planar polygons in 3-dimensional space, and a projection plane π, and wish to determine which portions of the polygonal boundaries are visible when viewed in a direction normal to π. We denote the set of polygons in S projected to π by S_π. For each point p in S we let $\pi(p)$ denote its projection in π, and for each point p in S_π we let $\pi^{-1}(p)$ denote the point in S which projects to p. Similarly, for each polygon P_i in S we let $\pi(P_i)$ denote its projection in S_π, and for each polygon P_i in S_π we let $\pi^{-1}(P_i)$ denote the polygon in S which projects to P_i. For any point p we let $x(p)$ and $y(p)$ denote the x- and y-coordinates of p, respectively. We present a high-level description of the algorithm below.

Step 1. Constructing the P-arrangement. We construct the P-arrangement of S_π in this step, as described above.

Step 2. Computing the coverage of representative vertices. In this step we use the P-arrangement to compute the *coverage*, c_i, of each representative vertex $rep(P_i)$. The coverage of a point p in S_π is defined to be the number of polygons in S which obscure $\pi^{-1}(p)$, i.e., the number of polygons intersected by the ray emanating from $\pi^{-1}(p)$ normal to π in the direction of the viewing "eye." We do this by traversing the P-arrangment in a depth-first search fashion, storing the names of all the polygons which contain our current position as we go. Each coverage computation is done by examining the polygons in this set. The total time for this traversal is $O(n+k+t)$.

Step 3. Computing visible edges. In this step we use the P-arrangement to "walk" around the boundary of each polygon P_i, starting at its representative vertex, computing the coverage of each edge portion as we go. All the portions of ∂P_i which have zero coverage are marked "visible" (or displayed). This takes $O(n+k)$ time and completes the algorithm.

End of High-Level Description.

We now describe in more detail how to perform each of the above steps. Since we have already shown how to perform Step 1, we begin with Step 2.

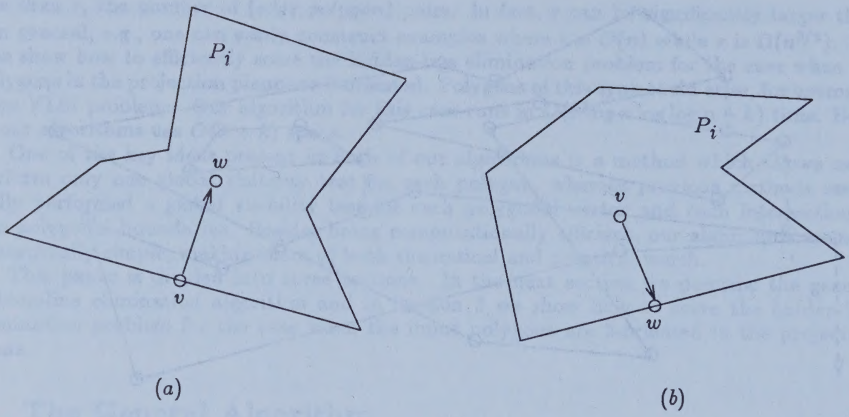


Figure 2: In going from v to w we are entering P_i in case (a) and are leaving P_i in case (b). Note: the edge (v, w) is undirected; the arrows in the figure simply denote the direction of movement.

2.2 Step 2: Computing the coverage of representative vertices.

We begin with some definitions. During any traversal of the P-arrangement, suppose we are currently at a node v and moving to a node w. We say that we are *entering* (respectively, *leaving*) the polygon P_i along (v, w) if v (resp., w) is contained in the boundary of P_i and the line segment associated with the edge (v, w) is properly contained in the interior of P_i. (See Figure 2.) We can detect whether we are leaving or entering a polygon in $O(1)$ time by examining the extra information we have stored with the vertices and edges in the P-arrangement.

Step 2 can be performed by traversing the P-arrangement of S_π in the following way. Let G denote the P-arrangement of S_π, and recall that R is the set of all representative vertices which do not have vertical shadows in S_π. Starting with w, the first vertex in R, we remove w from R and begin traversing G in a depth-first-search fashion starting with w. As we traverse we maintain a representation of the set D of polygons in S_π containing our current position. Since w is a leftmost vertex in S_π, there are no polygons which properly contain w in their interior. Thus, we begin with D being empty. We only store the name of each polygon in this list, along with a representation of the plane which contains the polygon. We begin the search with w, because there are no polygons which contain it. As we are traversing G, each time we enter the interior of a polygon (as in Figure 2.a) we insert it into D, and each time we leave the interior of a polygon (as in Figure 2.b) we delete it from D. Also, when we are "returning" along already traversed paths, i.e., popping off the stack of visitted nodes in the depth-first search, we reverse any operations we made along the way. Each time we encounter a representative vertex $rep(P_i)$ we examine all the polygons that are currently in D and count how many of them correspond to polygons in $\Re^3$ which obscure $\pi^{-1}(rep(P_i))$. Since we include vertical shadows as well as intersection points in G, any two intersecting polygons are connected in G, thus this counting procedure will always compute the coverage, c_i, of $rep(P_i)$. We then remove $rep(P_i)$ from R and continue our traversal. When we return to w, completing a traversal of the connected component of G containing w, we continue the depth-first search starting with the first vertex currently left in R. We repeat this traversal until R is empty. Since each edge in G belongs to some polygon in S_π, and every polygon initially has a representative vertex in R this method is guaranteed to visit every node in G.

Let us analyze the time complexity of this traversal. We can use a simple linked-list data structure to maintain D, the set of polygons containing our current position in the traversal. Since we are traversing the P-arrangement in a depth-first fashion, the set D will only change when we enter or leave a polygon, and then D only gains or loses one item.

Thus, each insertion and deletion can be performed in $O(1)$ time. The only other operation we perform in the traversal is computing the coverage of each representative vertex, which, for each *rep*(P_i), can be performed in time proportional to the number of polygons stored in D when we encounter *rep*(P_i). Since there is only one representative vertex for each polygon, and for each *rep*(P_i) we only examine those polygons which properly contain *rep*(P_i) in their interior, the number of comparisons we make for each P_i will be at most the number of polygons in S_π which intersect P_i. Thus, we make at most $O(t)$ comparisons overall, where t is the number of pairs of intersecting polygons in S_π. Since we are traversing the P-arrangement in a depth-first fashion, we will visit each node at most twice, and similarly for each edge. Thus, this step takes $O(n+k+t)$ time overall.

2.3 Step 3: Computing visible edges.

In this step we use the P-arrangement to "walk" around the boundary of each polygon P_i in S_π, in turn, starting with *rep*(P_i). We let $G(P_i)$ denote the subgraph of the P-arrangment G restricted to the boundary of P_i, i.e., the cycle of G corresponding to ∂P_i. We start with *rep*(P_i) and walk around $G(P_i)$, always maintaining the coverage of the point on P_i that corresponds to our current position in $G(P_i)$. We initialize a counter c to the value c_i, the coverage of *rep*(P_i) (which was computed in Step 2), and maintain the property that c is the coverage of our current position in the walk around $G(P_i)$ as follows: Let p be our current position in $G(P_i)$ and let q be the point we are moving to next in the traversal. If in going from p to q we enter a polygon P_j such that $\pi^{-1}(P_j)$ obscures $\pi^{-1}(q)$, we increment the counter c, and if we leave a polygon P_j such that $\pi^{-1}(P_j)$ obscures $\pi^{-1}(p)$, we decrement c. Any edge portions of $G(P_i)$ we traverse with $c=0$ we mark as being "visible." Any vertices p of $G(P_i)$ we traverse with $c=0$, or such that c is being incremented from 0 or decremented to 0 at p, we also mark as being "visible." This gives us all the visible portions of ∂P_i. The total time needed to traverse all the polygons in S_π in this manner is $O(n+k)$, since we traverse each vertex once and each intersection point twice. This completes the algorithm.

We summarize this section in the following theorem.

Theorem 2.2 *Given a set S of simple, planar polygons in $\Re^3$, consisting of n total edges, and a projection plane π, the hidden-line elimination problem for S can be solved in $O(n\log n + k+t)$ time and $O(n+k)$ space, where k (resp. t) is the number of intersecting pairs of edges (polygons) in S_π.* ∎

In the next section we study an important special case of hidden-line elimination.

3 Polygons which are 2-Oriented in π

In this section we study the case when the input polygons are 2-oriented in the projection plane π. Polygons of this type could arise, for example, from VLSI problems. In this section we show how to perform hidden-line elimination for this case in $O(n\log n+k)$ worst-case time and $O(n+k)$ space. Without loss of generality, we assume each edge is either parallel to the x-axis or to the y-axis, that is, the polygons are rectilinear. One can apply a simple transformation to modify our algorithm to the general case. Suppose we are given a collection S of polygons in $\Re^3$ which project to rectilinear polygons in the projection plane π. We begin with a high-level description of the algorithm:

Step 1. Constructing the P-Arrangement. In this step we construct the P-arrangement as before. Let G denote the P-arrangement of S_π. As before, we also let the representative vertex of each polygon P_i, denoted *rep*(P_i), be the leftmost vertex in P_i. Since the edges of the polygons in π are rectilinear, this step can easily be done in $O(n\log n+k)$ time and $O(n+k)$ space using the algorithm by Bentley and Ottmann [2].

Step 2. Computing Restricted Visibility. In this step we compute the visible portions of each polygon P_i when visibility is restricted to the set, Σ_i, of polygons whose boundary intersects the boundary of P_i. We define the *coverage* of a point p in S_π *with respect to* a subset, Σ, of polygons from S_π to be number of polygons P_i in Σ such that $\pi^{-1}(P_i)$ obscures $\pi^{-1}(p)$, i.e., the coverage p would have if $S_\pi=\Sigma$. For each polygon P_i, we perform this step by using the P-arrangement data structure to "walk" around the cycle $G(P_i)$ of G corresponding to perimeter of P_i twice: the first time to compute the coverage of $rep(P_i)$ with respect to Σ_i, and the second time to compute the visible portions of P_i when we restrict our attention to the polygons in Σ_i (i.e., we ignore all the other polygons in S_π).

Step 3. The Clean-Up Step: Removing Incorrectly Marked "Visible" Portions. In performing Step 2 it is possible that we marked a portion of a polygon P_i as being "visible" when, in fact, it is not. This is because for any polygon P_i with part of its boundary marked "visible" in Step 2, there may be a polygon P_j which obscures P_i and whose boundary does not intersect the boundary of P_i. In this step we remove the polygonal portions mistakenly marked as being "visible" in Step 2. We perform this by reducing it to an equivalent 2-dimensional rectangle searching problem. This problem can then be solved in $O(n \log n \log\log n)$ time and $O(n)$ space.

End of High-Level Description.

Let us now concentrate on the details of how to implement each of the above steps. Since we have already described how to construct the P-arrangement (Step 1), we need only to describe Steps 2 and 3.

3.1 Step 2: Computing Restricted Visibility

In this step we need to compute all the visible portions of each polygon P_i when visibility is restricted to the set Σ_i, the set of polygons whose boundary intersects the boundary of P_i. Our method is to use the P-arrangement to walk around $G(P_i)$ twice. In the first walk we compute the coverage of the representative vertex $rep(P_i)$ with respect to Σ_i. We do this as follows. Starting with $rep(P_i)$ we walk around $G(P_i)$, maintaining a list of polygons which contain our current position, represented as as a simple linked list D. When we enter a polygon we insert its name in D, and when we leave a polygon we delete its name from D, provided it is already in the list. If we leave a polygon that is not already in D, we do nothing. By the Jordan Curve Theorem, after walking around a polygon P_i in this manner, starting at $rep(P_i)$ and returning to $rep(P_i)$, D will store all the polygons in Σ_i which contain $rep(P_i)$ in their interior. Upon returning to $rep(P_i)$, we examine D and count how many polygons P_j are in D such that $\pi^{-1}(P_j)$ obscures $\pi^{-1}(p)$, thus computing c_i, the coverage of $rep(P_i)$ with respect to Σ_i. We then perform a second walk around the polygon P_i, which resembles the walk of Step 3 in the algorithm of the previous section. Let c be initialized to c_i, the coverage of $rep(P_i)$ with respect to Σ_i. In the second walk around P_i, whenever we enter a polygon corresponding to a polygon which obscures the point corresponding to our current position we increment the counter c, and whenever we leave a polygon corresponds to a polygon which corresponds to a polygon which obscures the point corresponding to our previous position we decrement c. Any portions of P_i we traverse with $c = 0$ we mark as being "visible." This gives us all the visible portions of P_i when we restrict our attention to the polygons in Σ_i. If we found a "visible" portion of P_i, then we define a distinguished representative point of P_i, denoted $vis(P_i)$, to be equal to the leftmost "visible" point of P_i. If we never find a visible part of P_i, then we define $vis(P_i)$ to be $\emptyset$. The total time needed to traverse all the polygons in S in this manner is $O(n + k)$, and we need $O(n + k)$ space to store the P-arrangement.

3.2 Step 3: The Clean-Up Step

After performing Step 2 we know that the only way we could have mistakenly marked a portion of a polygon P_i as "visible" is if $\pi^{-1}(P_i)$ is completely obscured by another polygon in S. Thus, for every polygon P_i, since we distinguished a point $vis(P_i)$ in one of the portions of P_i marked "visible" (if there were any such portions), it is sufficient to determine which of the distinguished $vis(P_i)$ points are really visible. After we know which of the $vis(P_i)$ points are really visible, we can then, for each $vis(P_i)$, output all the edges on P_i marked "visible," since they are in fact visible edges. The edges we do not output are the hidden-lines. Given which of the $vis(P_i)$'s are really visible, this can easily be done in $O(n + k)$ additional time. In short, we have reduced a polygon-visibility problem to a point-visibility problem. It is this point-visibility problem that we solve below.

The main idea of our method is to further reduce the point-visibility problem to an equivalent 2-dimensional rectangle searching problem. Specifically, we reduce it to the following problem: given a set $R = \{r_1, r_2, \ldots, r_n\}$ of rectangles and a set $W = \{p_1, p_2, \ldots, p_m\}$ of points (all in the plane), compute for each point $p_i \in W$ the value of the function $SUM(p_i) = \sum_{r_j \in R(p_i)} val(r_j)$, where $R(p_i)$ denotes the set of all rectangles in R which contain p_i, $val(r_j)$ is a pre-computed value associated with the rectangle r_j, and $+$ defines a commutative semigroup. The reduction is as follows. For each polygon $P_i \in S_\pi$ use horizontal line segments to partition P_i into rectangles with sides parallel to the x- and y-axes. We can perform this for each P_i by a simple plane-sweeping procedure, partitioning P_i into a number of rectangles proportional to the number of edges on the boundary of P_i. This takes $O(n \log n)$ time and $O(n)$ space to partition all the polygons in S_π. Let R be the set of all rectangles defined in this way from polygons in S_π. For any rectangle r in π let $\Pi(r)$ denote the 3-dimensional solid $r \times (-\infty, +\infty)$. For each r_j in R, we define $val(r_j)$ to be $\Pi(r_j) \cap \pi^{-1}(P_i)$, where P_i is the polygon which was partitioned to form r_j. In other words, the value of r_j is the quadrilateral in S which projects to r_j. Given two non-intersecting quadrilaterals q_1 and q_2 which project to rectangles r_1 and r_2, respectively, in π, define $q_1 + q_2$ as follows:

$$q_1 + q_2 = \begin{cases} \emptyset & \text{if } r_1 \cap r_2 = \emptyset \\ \Pi(r_1 \cap r_2) \cap q_1 & \text{if } q_1 \text{ obscures some part of } q_2 \\ \Pi(r_1 \cap r_2) \cap q_2 & \text{if } q_2 \text{ obscures some part of } q_1 \end{cases}$$

Computationally speaking, we form $q_1 + q_2$ by determining if they project to intersecting rectangles r_1 and r_2 in π, and, if so, take their sum to be the visible quadrilateral (on q_1 or q_2) which projects to $r_1 \cap r_2$. Note that this defines a commutative semigroup, since (i) the polygons in S do not intersect in $\Re^3$, (ii) the intersection of two rectangles is always a rectangle (or is empty), and (iii) $+$, as defined above, is associative and commutative. Also notice that $q_1 + q_2$ can be computed in $O(1)$ time. As the final step in the reduction, we take W to be the set of all the "visible" points $vis(P_i)$.

Unfortunately, we cannot use the data structures of Gabow et al. [8] or Willard [19], which would run in $O(n \log n)$ time for this rectangle searching problem, because they are only defined for groups, not semigroups, i.e., they assume inverses exist. Consequently, we need a different approach. Our method is based on a plane-sweeping procedure using a semi-dynamic 1-dimensional data structure of Hirschberg and Volper [11]. For expository (and space) reasons, we describe the procedure so as to have a running time which is $O(n \log n \log \log n)$ and so as to use $O(n \log n \log \log n)$ space. In the technical report version of this paper [9] we show how to apply the "streaming" paradigm of Edelsbrunner and Overmars [7] to this plane-sweep procedure to bring the space complexity down to $O(n)$.

The main idea is to solve the rectangle searching problem by sweeping a horizontal line L through the plane π from top to bottom, much like the scan-line on a CRT screen. We will maintain a set K, which will correspond to the set of intervals which are formed by the intersections of the sweep-line L with the polygons in S_π. Since the polygons in S_π are rectilinear and L is a horizontal line, the only time that K can change is at horizontal boundary segments. Thus, we define the *event* list for the plane-sweep to be the list of horizontal line segments which belong to the boundaries of the (partition) rectangles in π, sorted by decreasing y-coordinates. The plane-sweep will be implemented by starting with the first line segment in the event list and iteratively removing segments from the front of this list, performing the approapriate operations for each segment so as to maintain K as defined above.

We maintain a data structure, denoted $Semi(K)$, which is the data structure by Hirshberg and Volper [11], to store the values associated with the rectangles currently "stabbed" by the sweep-line L. Let us briefly review this data structure. Given a set of intervals I, with endpoints in the range of integers from 1 to n, and such that there is a value associated with each interval from a commutative semigroup, it allows the following two operations: (i) **update**(I_j, w): change the value of the interval I_j to w, and (ii) **query**(k): compute the summation of the values belonging to all the intervals which contain k. Their data structure uses $O(n \log n \log \log n)$ space, requires at most $O(n \log n \log \log n)$ preprocessing time, allows for updates to be performed in $O(\log n \log \log n)$ time, and queries to be performed in $O(\log n)$ time. We initalize $Semi(K)$ so that it stores all the intervals in the event list, and we associate the value of the identity $r_{-\infty}$ with each one. The semigroup identity $r_{-\infty}$ is simply the back clip rectangle, i.e., the rectangle which projects to the view window in π and is below all the rectangles in R. (The interested reader may wish to verify that $r_{-\infty}$ is in fact an identity element on the semigroup defined by the + operation.) As an important implementation detail, the data structure of Hirshberg and Volper is only defined for integer endpoints. To get around this restriction, we map the line segments in the event list to *rank space*. That is, we sort the endpoints of the intervals in the event list by increasing x-coordinates and take the integer name of each one to be its rank in this sorted list. This can be done as a part of the preprocessing for $Semi(K)$, as it requires at most $O(n \log n)$ time. Mapping geometric objects to integers in this manner is a technique common to a number of computational geometry algorithms, e.g., see [3,8].

The plane-sweep proceeds as follows. At each event in the sweep we remove a horizontal segment s from the front of the event list. For each "visible" point $vis(P_i)$ on s, if any, we find the integer k corresponding to $vis(P_i)$ in the endpoint rank space, and perform a **query**(k) operation on $Semi(K)$. If the value returned by the query is a quadrilateral which is below $\pi^{-1}(vis(P_i))$ or one that contains $\pi^{-1}(vis(P_i))$, then $vis(P_i)$ is really visible. If the value returned by the query is a quadrilateral which is above $\pi^{-1}(vis(P_i))$, however, then $vis(P_i)$ is not visible after all; hence, P_i is completely obscured by another polygon. We complete the operations for segment s by updating the value of the interval in $Semi(K)$ which corresponds to s. Let I_s be the interval in K corresponding to s. If s is the bottom edge of a partition rectangle, we update I_s's value to be $r_{-\infty}$, effectively removing I_s from $Semi(K)$. If s is the top edge of a partition rectangle, then we update I_s's value to be the value of that rectangle, effectively inserting I_s in $Semi(K)$. As a minor optimization, we could check if s is the top and bottom edge of two rectangles in the same polygon, in which case we need not make any changes to $Semi(K)$ for s. When we complete the plane-sweep we will determined for each "visible" point $vis(P_i)$ whether $vis(P_i)$ is really visible or not. This completes the reduction.

For each segment s in the event list we spend $O(\log n \log\log n)$ time to update the value of the interval corresponding to s in $Semi(K)$. Also, for each "visible" point $vis(P_i)$ we encounter during the plane-sweep, we spend $O(\log n)$ time to perform a query for $vis(P_i)$. Since there are $O(n)$ horizontal segments in the event list, and $O(n)$ "visible" points for which we performed queries, the entire plane-sweep can be performed in $O(n \log n \log\log n)$ time and $O(n \log n \log\log n)$ space. In the technical report [9] we show how to apply the "streaming" pradigm of Edesbrunner and Overmars [7] to this reduction to reduce the space compexity to $O(n)$. We summarize with the following theorem.

Theorem 3.1 *Suppose one is given a set S of polygons in $\Re^3$ and a projection plane π. If the polygons in S project to 2-oriented polygons in π, one can solve the hidden-line elimination problem for S in $O(n \log n \log\log n + k)$ time and $O(n + k)$ space.* ∎

4 Conclusion

In this paper we gave algorithms for hidden-line elimination which are optimal in the worst case and are also able to take advantage of problem instances which are "simpler" than in the worst case. Our approach of using the P-arrangement is an example of the paradigm from computational geometry of reducing geometric problems to graph problems. Our algorithm for solving the general problem ran in $O(n \log n + k + t)$ time, where n is the number of edges and k (resp. t) is the number of intersecting pairs of edges (resp. polygons) in the projection plane. If the polygons in the projection plane are 2-oriented, then we showed how to solve the hidden-line problem in $O(n \log n \log\log n + k)$ time. Both of our algorithms used $O(n+k)$ space. They should also be of practical worth, because of their conceptual simplicity. Incidentally, it is easy to extend all of our results to allow for polygonal holes in the input polygons.

There are a number of other problems that can be solved efficiently using the P-arrangment. For example, given a collection S of polygons in the plane, computing the contour of the union of the polygon in S, reporting all intersecting pairs of polygons in S, and finding a point contained in a maximum number of polygons in S are all such problems. We leave the details of these constructions to the reader.

Acknowledgements

We would like to thank Mikhail Atallah, Joseph O'Rourke, and Michael McKenna for several helpful conversations.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley (Reading, Mass.: 1974).

[2] J.L. Bentley and T.A. Ottmann, "Algorithms for Reporting and Counting Geometric Intersections," *IEEE Trans. on Computers*, Vol. C-28, No. 9, September 1979, 643–647.

[3] J.L. Bentley and D. Wood, "An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles," *IEEE Trans. on Computers*, Vol. C-29, No. 7, July 1980, 571–576.

[4] B. Chazelle and H. Edelsbrunner, Personal communication, 1987.

[5] F. Dévai, "Quadratic Bounds for Hidden-Line Elimination," *Proc. 2nd ACM Symp. on Computational Geometry*, 1986, 269–275.

[6] H. Edelsbrunner, M.H. Overmars, and R. Seidel, "Some Methods of Computational Geometry Applied to Computer Graphics," *Computer Vision, Graphics, and Image Processing*, Vol. 28, 1984, 92–108.

[7] H. Edelsbrunner and M.H. Overmars, "Batched Dynamic Solutions to Decomposable Searching Problems," *Journal of Algorithms*, Vol. 6, 1985, 515–542.

[8] H. Gabow, J. Bentley, and R. Tarjan, "Scaling and Related Technique for Geometry Problems," *Proc. of 16th ACM Symp. on Theory of Computing*, 1984, 135–143.

[9] M.T. Goodrich, "A Polygonal Approach to Hidden-Line Elimination," Tech. Report JHU 87-18, Dept. of Computer Science, The Johns Hopkins University, 1987.

[10] R.H. Güting and T. Ottmann, "New Algorithms For Special Cases of the Hidden Line Elimination Problem," *Proc. Symp. on Theoretical Aspects of Computer Science*, 1985, 161–171.

[11] D.S. Hirschberg and D.J. Volper, "Improved Update/Query Algorithms for the Interval Valuation Problem," *Information Processing Letters*, Vol. 24, March 1987, 307–310.

[12] M. McKenna, "Worst-case Optimal Hidden-Surface Removal," Tech. Report JHU/EECS-86/05, Dept. of EECS, Johns Hopkins Univ., 1986.

[13] O. Nurmi, "A Fast Line-Sweep Algorithm For Hidden Line Elimination," *BIT*, Vol. 25, 1985, 466–472.

[14] T. Ottmann and P. Widmayer, "Solving Visibility Problems by Using Skeleton Structures," *Proc. 11th Symp. on Mathematical Foundations of Computer Science*, 1984, 459–470.

[15] A. Schmitt, "On the Time and Space Complexity of Certain Exact Hidden Line Algorithms," Universität Karlsruhe, Fakultät für Informatik, Report 24/81, 1981. Cited in [10].

[16] A. Schmitt, "Time and Space Bounds for Hidden Line and Hidden Surface Algorithms," *EUROGRAPHICS '81*, 43–56.

[17] S. Sechrest and D.P. Greenberg, "A Visibility Polygon Reconstruction Algorithm," *ACM Transactions on Graphics*, Vol. 1, No. 1, January 1982, 25–42.

[18] I.E. Sutherland, R.F. Sproull, and R.A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys*, Vol. 6, No. 1, March 1974, 1–25.

[19] D.E. Willard, "New Data Structures for Orthogonal Range Queries," *SIAM Journal on Computing*, Vol. 14, No. 1, Feb. 1985, 232–253.

A DUAL APPROACH TO DETECT POLYHEDRAL INTERSECTIONS IN ARBITRARY DIMENSIONS†

OLIVER GUNTHER and EUGENE WONG
EECS Department, 231 Cory Hall
University of California
Berkeley CA 94720
gunther@postgres.berkeley.edu

Abstract

This paper presents a dual approach to detect intersections of hyperplanes and convex polyhedra in arbitrary dimensions. In d dimensions, the time complexities of the dual algorithms are $O(2^d d \log n)$ for the hyperplane-polyhedron intersection problem, and $O((2d)^{d-1} \log^{d-1} n)$ for the polyhedron-polyhedron intersection problem. In two dimensions, these time bounds are achieved with linear space and preprocessing. In three dimensions, the hyperplane-polyhedron intersection problem is also solved with linear space and preprocessing; quadratic space and preprocessing, however, is required for the polyhedron-polyhedron intersection problem. For general d, the dual algorithms require $O(n^{2^{d-2}d})$ space and $O(2^d n^{2^{d-2}d} \log n)$ preprocessing. All of these results readily extend to unbounded polyhedra.

1. Introduction

Detecting and computing intersections is a fundamental problem in computational geometry [Lee84]. Fast solutions for intersection problems are desirable in a wide range of application areas, including linear programming [Dant63] , hidden surface elimination [Newm79] , or spatial databases [Gunt87]. In many of these applications, the dimension of the intersection problems may be greater than three. This is particularly obvious in linear programming; another example are database applications where geometric objects are used to represent predicates [Ston86].

It was first noted by Chazelle and Dobkin [Chaz80] that it is often easier to *detect* the intersection of two suitably preprocessed geometric objects rather than to actually *compute* it. In the detection problem, one only asks if two objects intersect or not; also, it is allowed to preprocess each of the given objects separately.

In this paper, we present algorithms to solve the intersection detection problem in arbitrary dimensions for hyperplanes and convex polyhedra. A *(d-dimensional, convex) polyhedron* P in d-dimensional Euclidean space E^d is defined to be the intersection of some finite number of closed halfspaces in E^d, such that the dimension of the smallest affine subspace containing P is d. If $a \in E^d-\{0\}$ and $c \in E^1$ then the $(d-1)$-dimensional set $H(a,c)=\{x \in E^d : x \cdot a = c\}$ is called a *hyperplane* in E^d.* A hyperplane $H(a,c)$ defines two closed halfspaces $H^+(a,c)=\{x \in E^d : x \cdot a \geq c\}$ and $H^-(a,c)=\{x \in E^d : x \cdot a \leq c\}$. A hyperplane $H(a,c)$ *supports* a polyhedron P if $H(a,c) \cap P \neq \phi$ and $P \subseteq H^+(a,c)$. If $H(a,c)$ is any hyperplane supporting P then $P \cap H(a,c)$ is a *face* of P. The faces of dimension 1 are called *edges*; those of dimension 0 *vertices*. A supporting hyperplane is called a *boundary* hyperplane is the face $H(a,c) \cap P$ is of dimension $d-1$. The faces of P that are a subset of some supporting hyperplane $H(a,c)$, with $a_d \leq 0$, form the *upper hull* of P. The *lower hull* of P is defined similarly.

So far, the intersection detection problem has only been considered in two and three dimensions. In their original paper, Chazelle and Dobkin [Chaz80] solve the d-dimensional hyperplane-polyhedron intersection problem in time $O(\log n)$ $(d=2)$ and $O(\log^2 n)$ $(d=3)$, and the polyhedron-polyhedron intersection problem in time $O(\log n)$ $(d=2)$ and $O(\log^3 n)$ $(d=3)$. Here, n denotes the maximum number of vertices of any given polyhedron. Both problems require $O(n)$ $(d=2)$ and $O(n^2)$ $(d=3)$ space and preprocessing. A revised version of that paper has been published recently [Chaz87]. In the three-dimensional case,

---

† This research was sponsored under grant DAAG 29-85-0223 and an IBM Fellowship for the first author.

* $x \cdot y$ denotes the inner product of vectors x and y.

$O(n \log n)$ space and preprocessing are also sufficient [Dobk80], in which case the running times given above have to be multiplied by a $\log n$ factor.

In a later paper, Dobkin and Kirkpatrick [Dobk83] improve the running times of Chazelle and Dobkin for the three-dimensional case by a factor of $\log n$. The new upper bounds are $O(\log n)$ and $O(\log^2 n)$ for the hyperplane-polyhedron and the polyhedron-polyhedron problems, respectively. As the algorithms of Chazelle and Dobkin, their algorithms require $O(n^2)$ storage and preprocessing. Again, the results of Dobkin and Munro [Dobk80] can be used to reduce the space and preprocessing requirements in three dimensions to $O(n \log n)$, in which case the running times increase by a $\log n$ factor.

In d dimensions, we obtain upper time bounds of $O(2^d d \log n)$ to detect the intersection of a hyperplane and a polyhedron, and $O((2d)^{d-1} \log^{d-1} n)$ to detect the intersection of two polyhedra. These time bounds appear to be the first results for $d>3$ and match the time bounds given by Dobkin and Kirkpatrick [Dobk83] for $d=2$ and $d=3$. Furthermore, our results seem to be the first of their kind that extend to unbounded polyhedra as well.

We obtain our results by means of a geometric duality transformation in d-dimensional Euclidean space $\mathbf{E}^d$ that is an isomorphism between points and hyperplanes [Brow79, Lee84]. Each convex polyhedron P is represented by a set of two functions in the dual space, TOP^P, BOT^P: $\mathbf{E}^{d-1} \rightarrow \mathbf{E}^1$, such that a hyperplane h intersects P if and only if the dual of h lies between TOP^P and BOT^P. Then, two polyhedra P and Q intersect if and only if for all $x \in \mathbf{E}^{d-1}$, we have $TOP^P(x) \geq BOT^Q(x)$ and $TOP^Q(x) \geq BOT^P(x)$.

For $d=2$ and for the hyperplane-polyhedron intersection problem in $d=3$, the space and preprocessing requirements of the dual representation scheme are $O(n)$ and therefore optimal. For the three-dimensional hyperplane-polyhedron intersection problem, this represents an improvement over the results of Dobkin and Kirkpatrick [Dobk83] by a factor of n. The three-dimensional polyhedron-polyhedron problem takes quadratic space and preprocessing, as does the algorithm of Dobkin and Kirkpatrick.

For general d, the scheme requires $O(n^{2^{d-2}d})$ space and $O(2^d n^{2^{d-2}d} \log n)$ preprocessing. To improve these bounds is a subject of further research. In particular, we suspect that lower bounds may be achieved at the expense of slightly higher time bounds for the detection algorithms.

Section 2 introduces the dual representation scheme for convex polyhedra. Sections 3 and 4 show how the hyperplane-polyhedron and the polyhedron-polyhedron intersection detection problems can be solved efficiently using the dual scheme. Section 5 presents several extensions of our approach, and section 6 contains our conclusions.

2. The Dual Representation Scheme

Let h denote some non-vertical $(d-1)$-dimensional hyperplane in $\mathbf{E}^d$. That is, in a d-dimensional Cartesian coordinate system, h intersects the d-th coordinate axis in a unique and finite point and can be represented by an equation

$$x_d = a_1 x_1 + \ldots + a_{d-1} x_{d-1} + a_d .$$

F_h denotes the function whose graph is h, i.e.

$$F_h : \mathbf{E}^{d-1} \rightarrow \mathbf{E}^1$$

$$F_h(x_1 \ldots x_{d-1}) = a_1 x_1 + \ldots + a_{d-1} x_{d-1} + a_d .$$

A point $p = (p_1 \ldots p_d)$ lies *above* (*on*, *below*) h if $p_d > (=,<) F_h(p_1 \ldots p_{d-1})$.

Brown [Brow79] defines a duality transformation D in $\mathbf{E}^d$ that maps hyperplanes into points and vice versa. The dual $D(h)$ of hyperplane h is the point $(a_1 \ldots a_d)$ in $\mathbf{E}^d$. Conversely, the dual $D(p)$ of a point p is the hyperplane defined by the equation

$$x_d = -p_1 x_1 - p_2 x_2 - \ldots - p_{d-1} x_{d-1} + p_d .$$

Lemma 2.1: A point p lies above (on, below) a hyperplane h if and only if the dual $D(h)$ lies below (on, above) $D(p)$.

Proof: Let h be given by the equation $F_h(x_1 \ldots x_{d-1}) = a_1 x_1 + \ldots + a_{d-1} x_{d-1} + a_d$ and let $p = (p_1 \ldots p_d)$ be a point above (on, below) h, i.e.

$p_d > (=,<) F_h(p_1 \,..\, p_{d-1})$ (*).

Inserting $D(h)=(a_1 \,..\, a_d)$ into $F_{D(p)}$ yields

$F_{D(p)}(a_1 \,..\, a_{d-1}) = -p_1a_1- \,..\, -p_{d-1}a_{d-1}+p_d > (=,<)\, a_d$ (due to (*))

Hence, $D(p)$ lies below (on, above) $D(h)$. □

A hyperplane h intersects a bounded polyhedron P if and only if there are two vertices v and w of P such that h lies between v and w (i.e. v lies on or above h and w lies on or below h, or vice versa). According to lemma 2.1, this is the case if and only if the dual $D(h)$ lies between the duals $D(v)$ and $D(w)$.

This observation leads to a new representation scheme for bounded convex polyhedra. Consider the functions $TOP^P, BOT^P : E^{d-1} \rightarrow E^1$ that are defined for a convex polyhedron P as follows. Here, V_P denotes the set of vertices of P.

$$TOP^P(x_1 \,..\, x_{d-1}) = \max_{v \in V_P} F_{D(v)}(x_1 \,..\, x_{d-1})$$

$$BOT^P(x_1 \,..\, x_{d-1}) = \min_{v \in V_P} F_{D(v)}(x_1 \,..\, x_{d-1})$$

Obviously, both functions are piecewise linear, continuous, and TOP^P is convex, whereas BOT^P is concave [Rock70]. With this notation, a non-vertical hyperplane h intersects P if and only if $D(h)$ lies between TOP^P and BOT^P. More formally, the hyperplane h, given by the equation $x_d = a_1x_1 + \,..\, + a_{d-1}x_{d-1} + a_d$, intersects P if and only if $BOT^P(a_1 \,..\, a_{d-1}) \le a_d \le TOP^P(a_1 \,..\, a_{d-1})$. A two-dimensional example of a polyhedron P and the corresponding functions TOP^P and BOT^P is given in figure 2.1.

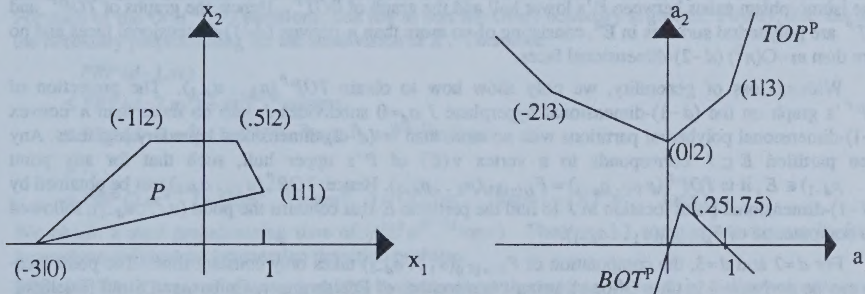


Figure 2.1

It is easily possible to extend this representation scheme to unbounded polyhedra. For simplicity, however, the main part of this paper is restricted to bounded polyhedra; the case of unbounded polyhedra is discussed in more detail in section 5.1.

The two functions TOP^P and BOT^P can be viewed as a mapping that map any slope $(a_1 \,..\, a_{d-1})$ of a non-vertical hyperplane into the maximum (TOP^P) or minimum (BOT^P) intercept a_d such that the hyperplane given by $x_d = a_1x_1 + \,..\, + a_{d-1}x_{d-1} + a_d$ intersects the polyhedron. We have

Theorem 2.2: Each convex polyhedron P corresponds to exactly one pair of functions (TOP^P, BOT^P), and conversely.

Proof: The functions TOP^P and BOT^P are uniquely defined for any convex regular polyhedron P, i.e. there is only one pair of functions (TOP^P, BOT^P) for any P.

Conversely, suppose there were two convex polyhedra P and Q such that $P \neq Q$, but $TOP^P(x_1 \,..\, x_{d-1}) = TOP^Q(x_1 \,..\, x_{d-1})$ and $BOT^P(x_1 \,..\, x_{d-1}) = BOT^Q(x_1 \,..\, x_{d-1})$ for all $(x_1 \,..\, x_{d-1}) \in E^{d-1}$.

Case 1: $P \cap Q = \phi$. Then there exists a non-vertical separating hyperplane h such that all points of P lie above h and all points of Q lie below h, or vice versa. There also exists a hyperplane h' parallel to h that intersects P. h' does not intersect Q. I.e., the dual $D(h')$ lies between TOP^P and BOT^P, but not between TOP^Q and BOT^Q. This is a contradiction to our assumption.

Case 2: $P \cap Q \neq \phi$. Because of $P \neq Q$ it is $P-Q \neq \phi$ or $Q-P \neq \phi$. W.l.o.g., let $P-Q \neq \phi$. Let p be some interior point of $P-Q$. There exists a non-vertical separating hyperplane h such that all points of Q lie above h and point p lies below h, or vice versa. There also exists a hyperplane h' parallel to h that goes through p. Because of $p \in P$, h' intersects P, but it does not intersect Q. Contradiction to our assumption as above. □

3. Hyperplane-Polyhedron Intersection Detection

For simplicity of presentation, we assume that the given hyperplane is non-vertical. This can always be achieved by a suitable rotation of the coordinate system. It is also possible to extend our detection algorithm to detect intersections with a vertical hyperplane; see section 5.2 for details.

A non-vertical hyperplane h, given by $x_d = a_1x_1 + \ldots + a_{d-1}x_{d-1} + a_d$ intersects a bounded polyhedron P if and only if $BOT^P(a_1 \ldots a_{d-1}) \leq a_d \leq TOP^P(a_1 \ldots a_{d-1})$. Moreover, an intersecting hyperplane h supports P if and only if $a_d = BOT^P(a_1 \ldots a_{d-1})$ or $a_d = TOP^P(a_1 \ldots a_{d-1})$. Therefore, the intersection detection problem can be solved by obtaining the functional values $TOP^P(a_1 \ldots a_{d-1})$ and $BOT^P(a_1 \ldots a_{d-1})$. It follows from the definition of TOP^P and BOT^P that these values can be found in time $O(d \cdot n)$ by computing $F_{D(v)}(a_1 \ldots a_{d-1})$ for each vertex $v \in V_P$. With some preprocessing, however, one can obtain polylogarithmic time bounds as follows.

It follows from [Brow79] that there is the following isomorphism between the upper hull of the polyhedron P and the graph of TOP^P. Each k-dimensional face f of the upper hull of P corresponds to exactly one $(d-k-1)$-dimensional face $D(f)$ of TOP^P's graph, and vice versa. Furthermore, if two faces f_1 and f_2 of P's upper hull are adjacent, then so are the faces $D(f_1)$ and $D(f_2)$ of TOP^P's graph. The same isomorphism exists between P's lower hull and the graph of BOT^P. Hence, the graphs of TOP^P and BOT^P are polyhedral surfaces in E^d, consisting of no more than n convex $(d-1)$-dimensional faces and no more than $m = O(n^2)$ $(d-2)$-dimensional faces.

Without loss of generality, we only show how to obtain $TOP^P(a_1 \ldots a_{d-1})$. The projection of TOP^P's graph on the $(d-1)$-dimensional hyperplane $J: a_d = 0$ subdivides J into no more than n convex $(d-1)$-dimensional polyhedral partitions with no more than m $(d-2)$-dimensional boundary segments. Any given partition $E \subseteq J$ corresponds to a vertex $v(E)$ of P's upper hull, such that for any point $(p_1 \ldots p_{d-1}) \in E$, it is $TOP^P(p_1 \ldots p_{d-1}) = F_{D(v(E))}(p_1 \ldots p_{d-1})$. Hence, $TOP^P(a_1 \ldots a_{d-1})$ can be obtained by a $(d-1)$-dimensional point location in J to find the partition E that contains the point $(a_1 \ldots a_{d-1})$, followed by a computation of $F_{D(v(E))}(a_1 \ldots a_{d-1})$.

For $d=2$ and $d=3$, the computation of $F_{D(v(E))}(a_1 \ldots a_{d-1})$ takes only constant time. The point location can be performed in time $O(\log n)$, using the algorithm of Edelsbrunner, Guibas, and Stolfi [Edel86a]. The total time complexity to detect the intersection of a hyperplane and a polyhedron is therefore $O(\log n)$. The space and preprocessing requirements are only $O(n)$, due to the fact that, in our case, the given partitions are convex and therefore monotone.

For general d, it takes time $O(d)$ to compute the functional value $F_{D(v(E))}(a_1 \ldots a_{d-1})$. Dobkin and Lipton [Dobk76] solve a $(d-1)$-dimensional point location problem with m $(d-2)$-dimensional boundary segments recursively as follows. In a preprocessing step, they compute the $O(m^2)$ $(d-3)$-dimensional intersection segments formed by the m original boundary segments, and project them on some $(d-2)$-dimensional hyperplane K. This way, the point location problem can be solved by a point location problem in K, followed by a binary search of the m original segments. Therefore, the time complexity of the point location is

$$
\begin{aligned}
& TPL(d-1, m) \\
& \leq TPL(d-2, m^2) + (d-1)(\lfloor \log m \rfloor + 1) \\
& \leq \ldots \\
& \leq TPL(2, m^{2^{d-3}}) + \sum_{i=1}^{d-3} (d-i)\left(\left\lfloor 2^{i-1} \log m \right\rfloor + 1\right)
\end{aligned}
$$

$= O(2^d d \log m) = O(2^d d \log n)$

We obtain a total time complexity of $O(2^d d \log n)$.*

For general d, the space requirements of the dual algorithm are as follows. The equations of the $O(n)$ faces require space $O(dn)$. The space requirements to store a convex subdivision of E^2 with m boundary segments, $SP(2,m)$, is $O(m)$ [Edel86a]. For a subdivision of E^{d-1} with m boundary segments, one has to store a subdivision of the $(d-2)$-dimensional projection hyperplane K with m^2 boundary segments and a sequence of no more than m boundary segments for each of the partitions. The number of partitions is no more than $m^{2(d-2)}$ [Edel86b]. Therefore,

$$\begin{aligned}
&SP(d-1,m)\\
&\le SP(d-2,m^2) + m^{2(d-2)}m\\
&\le SP(d-3,m^4) + m^{4(d-3)}m^2 + m^{2(d-2)}m\\
&\le \ldots\\
&\le SP(2,m^{2^{d-3}}) + O(m^{2^{d-3}d}) = O(m^{2^{d-3}d}) = O(n^{2^{d-2}d}) .
\end{aligned}$$

We obtain a total space complexity of $O(n^{2^{d-2}d})$.

The preprocessing requirements of this algorithm are as follows. Each $(d-2)$-dimensional boundary segment of the subdivision is obtained from the original polyhedron P in time $O(d)$ by dualization and projection. Here, we assume that P is given by a list of its faces and the corresponding adjacency relations. As there are $m = O(n^2)$ $(d-2)$-dimensional boundary segments, it takes time $O(dn^2)$ to obtain all of them.

The preprocessing requirements to solve a point location problem in a convex subdivision of E^2 with m boundary segments, $PRP(2,m)$, are $O(m)$ [Edel86a]. For a subdivision of E^{d-1} with m boundary segments, one has to compute m^2 intersections, and to project them on some $(d-2)$-dimensional hyperplane K. For each of the $O(m^{2(d-2)})$ partitions, one has to sort the $O(m)$ boundary segments. Finally, one has to do the necessary preprocessing for the subdivision of K. Therefore,

$$\begin{aligned}
&PRP(d-1,m)\\
&\le PRP(d-2,m^2) + m^{2(d-2)}m \log m\\
&\le PRP(d-3,m^4) + m^{4(d-3)}m^2 \log m^2 + m^{2(d-2)}m \log m\\
&\le \ldots\\
&\le PRP(2,m^{2^{d-3}}) + O(m^{2^{d-3}d} \log m^{2^{d-4}}) = O(2^d m^{2^{d-3}d} \log m) = O(2^d n^{2^{d-2}d} \log n) .
\end{aligned}$$

We obtain a total preprocessing time of $O(2^d n^{2^{d-2}d} \log n)$. Theorem 3.1 summarizes our results for the hyperplane-polyhedron intersection detection problem.

Theorem 3.1: Given a non-vertical $(d-1)$-dimensional hyperplane h and a d-dimensional convex polyhedron P, h and P can be tested for intersection in time $T(d,n)$ with $S(d,n)$ space and $PP(d,n)$ preprocessing:

$P \cap h = \phi$?	$T(d,n)$	$S(d,n)$	$PP(d,n)$
$d=2$	$O(\log n)$	$O(n)$	$O(n)$
$d=3$	$O(\log n)$	$O(n)$	$O(n)$
$d>3$	$O(2^d d \log n)$	$O(n^{2^{d-2}d})$	$O(2^d n^{2^{d-2}d} \log n)$

Proof : follows from the preceding discussion. □

4. Polyhedron-Polyhedron Intersection Detection

Two convex polyhedra P and Q do not intersect if and only if there is a separating non-vertical hyperplane between them. Any such hyperplane h does not intersect either P or Q, but there are hyperplanes h' and h'' parallel to h, such that h' is above h and h'' is below h, and either h' intersects P and h'' intersects Q, or vice versa. More formally, a non-vertical hyperplane h, given by the equation $x_d = a_1 x_1 + \,.\,.\, + a_{d-1} x_{d-1} + a_d$, separates the polyhedra P and Q if and only if

---

* Note that we assume that it takes time $O(d)$ to determine on which side of a given hyperplane a point is located. Dobkin and Lipton [Dobk76] assume in their analysis that this can be done in constant time and consequently obtain a time complexity of $O(2^d \log n)$.

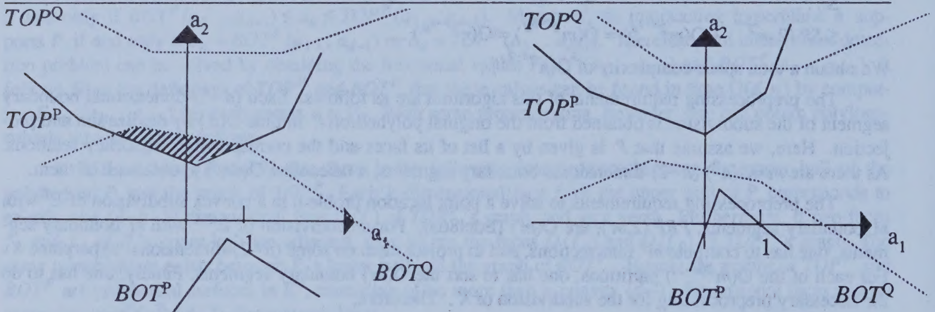
$$TOP^P(a_1 \,.\,.\, a_{d-1}) < a_d < BOT^Q(a_1 \,.\,.\, a_{d-1}), \text{ or}$$
$$TOP^Q(a_1 \,.\,.\, a_{d-1}) < a_d < BOT^P(a_1 \,.\,.\, a_{d-1}).$$

Therefore, two polyhedra P and Q intersect if and only if[*]

(i) $\min\limits_{(x_1 .. x_{d-1}) \in E^{d-1}} (TOP^P - BOT^Q)(x_1 \,.\,.\, x_{d-1}) \geq 0$, and

(ii) $\min\limits_{(x_1 .. x_{d-1}) \in E^{d-1}} (TOP^Q - BOT^P)(x_1 \,.\,.\, x_{d-1}) \geq 0$.

See figure 4.1 for two examples. If both conditions are only met as equalities, then only the boundaries of P and Q intersect, but not their interiors.



(a) no intersection: the points in the shaded area are the duals of the separating hyperplanes.

(b) intersection

Figure 4.1

With the definitions of TOP^P and BOT^P, these conditions form a linear programming problem with no more than $2n$ constraints. According to Megiddo [Megi84], the time complexity to solve this problem is bound by $2^{O(2^d)} \cdot 2n$. Hence, the conditions can be tested in linear time $O(n)$ if the dimension is fixed. With some preprocessing, however, the conditions can be tested in polylogarithmic time as follows.

Without loss of generality, we only show how to test condition (i). We present a multidimensional search technique that finds the minimum of a convex piecewise linear function in arbitrary dimensions. The technique is recursive; it solves a d-dimensional problem by solving $O(d \log n)$ $(d-1)$-dimensional problems, and so on.

In the two-dimensional case, condition (i) can be tested by a variation of Dobkin and Kirkpatrick's algorithm [Dobk83] to detect the intersection of two polygons. The graphs of TOP^P and BOT^Q are monotone convex polygonal chains with edges $t_1 \ldots t_k$ and $b_1 \,.\,.\, b_l$ $(k+l \leq 2n)$; see also figure 2.1. The relative position and the slopes of the edges $t_{\lfloor k/2 \rfloor}$ and $b_{\lfloor l/2 \rfloor}$ give enough information to eliminate half of the edges of one (or both) chains from further consideration without missing the minimum. The algorithm proceeds recursively, eliminating at least one quarter of the remaining edges at each recursion level. Therefore, the minimum is detected in time $O(\log n)$ without any preprocessing or extra space. A similar analysis yields the same bound to test condition (ii).

In order to solve the d-dimensional problem, we solve $O(d \log n)$ $(d-1)$-dimensional problems. It is well known [Dant63] that the global minimum of $TOP^P - BOT^Q$ occurs at some vertex of the graph of $TOP^P - BOT^Q$, i.e. at some vertex $M = (M_1 \,.\,.\, M_d)$ of TOP^P's graph TG or BOT^Q's graph BG. Let $(v_1 \,.\,.\, v_{|TG|})$ denote the sequence of vertices in V_{TG}, sorted by increasing x_1-coordinate. We consider the vertex $v_{\lfloor |TG|/2 \rfloor}$ and its x_1-coordinate $\overline{a}_1$, and compute the local minimum of TOP^P-BOT^Q along the hyperplane $x_1=\overline{a}_1$. This is a $(d-1)$-dimensional minimization problem and can be solved recursively; let $m=(m_1=\overline{a}_1,m_2, \ldots ,m_d)$ denote some point where the local minimum is assumed. Due to the convexity of TOP^P-BOT^Q, we can determine the position of M relative to m from the local slope of TOP^P-BOT^Q. We have

---

[*] We write $(f \pm g)(x)$ for $f(x) \pm g(x)$.

Lemma 4.1: It is $M_1>(<)m_1$ if and only if there is an $\varepsilon_0>0$, such that for all ε with $0<\varepsilon<\varepsilon_0$

$$\begin{array}{ll} & TOP^P-BOT^Q(m_1-\varepsilon,m_2 \ldots m_d) \\ >(<) & TOP^P-BOT^Q(m_1 \ldots m_d) \\ >(<) & TOP^P-BOT^Q(m_1+\varepsilon,m_2 \ldots m_d). \end{array}$$

Otherwise, m is a global minimum of TOP^P-BOT^Q.

Proof : Due to the convexity of the function TOP^P-BOT^Q, there is always an $\varepsilon_0>0$, such that for all ε with $0<\varepsilon<\varepsilon_0$ exactly one of the following conditions holds:

(i) $TOP^P-BOT^Q(m_1-\varepsilon,m_2 \ldots m_d) > TOP^P-BOT^Q(m_1 \ldots m_d) > TOP^P-BOT^Q(m_1+\varepsilon,m_2 \ldots m_d)$,

(ii) $TOP^P-BOT^Q(m_1-\varepsilon,m_2 \ldots m_d) < TOP^P-BOT^Q(m_1 \ldots m_d) < TOP^P-BOT^Q(m_1+\varepsilon,m_2 \ldots m_d)$,

(iii) $TOP^P-BOT^Q(m_1-\varepsilon,m_2 \ldots m_d) \geq TOP^P-BOT^Q(m_1 \ldots m_d)$
$\wedge$ $TOP^P-BOT^Q(m_1+\varepsilon,m_2 \ldots m_d) \geq TOP^P-BOT^Q(m_1 \ldots m_d)$

If condition (iii) holds, then m is a local minimum. Because TOP^P-BOT^Q is convex, m also has to be a global minimum. Conversely, if m is a global minimum, condition (iii) clearly has to be true.

We now show indirectly that $M_1>m_1$ implies condition (i). Suppose that $M_1>m_1$, but (i) does not hold. Because m is not a global minimum, condition (ii) has to be true. Let $r = (r_1,r_2=m_2,r_3, \ldots ,r_d)$ denote the minimum of TOP^P-BOT^Q along the hyperplane $x_2=m_2$. Due to (ii) and to the convexity of TOP^P-BOT^Q, it is $r_1<m_1$ and $r_d<m_d$. Therefore, the line segment (M,r) intersects the hyperplane $x_1=m_1$ in some point $s = (s_1=m_1,s_2, \ldots ,s_d)$. Because of $M_d \leq r_d$, it is $s_d \leq r_d$, and because of $r_d<m_d$, it is $s_d<m_d$. This is a contradiction, because s lies on the hyperplane $x_1=m_1$, and m is the minimum along this hyperplane. A two-dimensional example is given in figure 4.2.

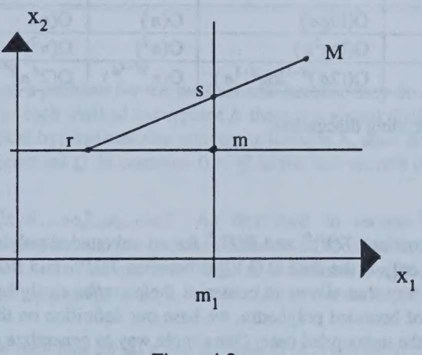


Figure 4.2

Hence, $M_1>m_1$ implies condition (i). Similarly, it can be shown that $M_1<m_1$ implies condition (ii). Due to the mutual exclusiveness of conditions (i), (ii) and (iii), we obtain that (i) implies $M_1>m_1$ and so on. This proves the lemma. ☐

Therefore, looking up the functional values $TOP^P-BOT^Q(m_1 \pm \varepsilon,m_2 \ldots m_d)$ for some suitable $\varepsilon>0$ gives us enough information to eliminate half of the vertices in V_{TG} (and some vertices in V_{BG}) from the search without missing the global minimum. If the search among the vertices in TG does not yield a global minimum, one continues with a similar search among the remaining vertices of BG. Hence the global minimum is obtained in no more than $\log(|TG|)+\log(|BG|)$ iterations.

The analysis of this algorithm obviously depends on the cardinalities of TG and BG. A simple combinatorial analysis shows that at any recursion level it is $|TG|+|BG|\le n^d$, i.e. the algorithm requires no more than $2d\log n$ iterations. Each iteration involves a $(d-1)$-dimensional minimization and the four function lookups necessary to obtain $TOP^P-BOT^Q(m_1\pm\varepsilon, m_2 \,..\, m_d)$. As shown in section 3, each lookup can be carried out in no more than $2^{d+1}d\log n$ steps. We obtain a total time complexity

$$\begin{aligned}
&T(d,n)\\
&\le d\log n\cdot(4\cdot 2^{d+1}d\log n+T(d-1,n))\\
&\le 2^{d+3}d^2\log^2 n+d\log n\,T(d-1,n)\\
&\le 2^{d+3}d^2\log^2 n+d\log n\,2^{d+2}(d-1)^2\log^2 n+d(d-1)\log^2 n\,T(d-2,n)\\
&\le \ldots\\
&\le \sum_{i=2}^{d-1} 2^{d+5-i}d^i\log^i n = O((2d)^{d-1}\log^{d-1} n)\,.
\end{aligned}$$

Of course, in practice one might be able to solve the intersection detection problem much faster by checking at various stages if $(TOP^P-BOT^Q)(x_1 \,..\, x_{d-1})<0$, or $(TOP^Q-BOT^P)(x_1 \,..\, x_{d-1})<0$.

For $d=3$, the space and preprocessing requirements of this algorithm are as follows. The equations of the $O(n)$ faces of P and Q require space $O(n)$. For the multidimensional binary search one has to store (a) a subdivision of the x_1-axis into no more than $n+1$ partitions, and (b) a sequence of $O(n)$ boundary segments for each one of the partitions. The total space requirements are therefore $O(n^2)$. The preprocessing can be done in time $O(n^2)$ by means of a plane sweep as described in [Prep85], pp. 47-48.

For $d>3$, the data structures required to do the search are essentially the same as the ones required to do the point location as described in section 3. Therefore, the space and preprocessing requirements are the same as for the hyperplane-polyhedron intersection detection problem. We obtain

Theorem 4.2: Given two d-dimensional convex polyhedra P and Q, P and Q can be tested for intersection in time $T(d,n)$ with $S(d,n)$ space and $PP(d,n)$ preprocessing:

$P\cap Q=\phi$?	$T(d,n)$	$S(d,n)$	$PP(d,n)$
$d=2$	$O(\log n)$	$O(n)$	$O(n)$
$d=3$	$O(\log^2 n)$	$O(n^2)$	$O(n^2)$
$d>3$	$O((2d)^{d-1}\log^{d-1} n)$	$O(n^{2^{d}-2d})$	$O(2^d n^{2^{d}-2d}\log n)$

Proof: follows from the preceding discussion. □

5. Extensions

5.1. Unbounded Polyhedra

Clearly, there exist functions TOP^P and BOT^P for an unbounded polyhedron P, such that a hyperplane h intersects P if and only if the dual $D(h)$ lies between TOP^P and BOT^P. The question is how to define these functions in a way that allows to construct their graphs easily by dualization of the original polyhedron P. In the case of bounded polyhedra, we base our definition on the notion of vertex, which is obviously not sufficient for the unbounded case. One simple way to generalize our definitions of TOP^P and BOT^P,

$$TOP^P(x_1 \,..\, x_{d-1}) = \max_{v\in V_P} F_{D(v)}(x_1 \,..\, x_{d-1})$$

$$BOT^P(x_1 \,..\, x_{d-1}) = \min_{v\in V_P} F_{D(v)}(x_1 \,..\, x_{d-1})$$

to an unbounded polyhedron P, is to enhance V_P by some *virtual* vertices at infinity. In particular, let C_P denote a d-dimensional cube with edge length $E(C_P)$ that contains all vertices of P. The bounded polyhedron $P\cap C_P$ has a set of vertices $V_{P\cap C_P}=V_P\cup\bar{V}$, where $\bar{V}$ contains those vertices that are formed by intersections of C_P with edges of P. As $E(C_P)$ goes to infinity, so do the vertices in $\bar{V}$. The dual $D(\bar{v})$ of any vertex $\bar{v}\in\bar{V}$ goes towards a vertical hyperplane with a corresponding function $F_{D(\bar{v})}:\mathrm{E}^{d-1}\rightarrow\pm\infty$. Now the functions $TOP^P, BOT^P:\mathrm{E}^{d-1}\rightarrow\mathrm{E}^1\cup\{\pm\infty\}$ are defined as

$$TOP^P(x_1 \, .. \, x_{d-1}) = \lim_{E(C_r) \to \infty} \max_{v \in V_r \cup \bar{V}} F_{D(v)}(x_1 \, .. \, x_{d-1})$$

$$BOT^P(x_1 \, .. \, x_{d-1}) = \lim_{E(C_r) \to \infty} \min_{v \in V_r \cup \bar{V}} F_{D(v)}(x_1 \, .. \, x_{d-1})$$

Again, there is an isomorphism between the upper hull of P and the graph of TOP^P, as well as between the lower hull of P and the graph of BOT^P [Brow79]. Note that the virtual vertices are only a conceptual aid. They do not have to be taken into account when constructing the graphs by dualization. If the dual of P's upper hull does not yield a finite value $a_d = TOP^P(a_1 \, .. \, a_{d-1})$, then the functional value at $(a_1 \, .. \, a_{d-1})$ is assumed $+\infty$. Similarly, the default for $BOT^P(a_1 \, .. \, a_{d-1})$ is $-\infty$. The algorithms to detect intersections do not have to be modified, except for the possibility that TOP^P and BOT^P may now assume the values $\pm\infty$. A two-dimensional example is given in figure 5.1.

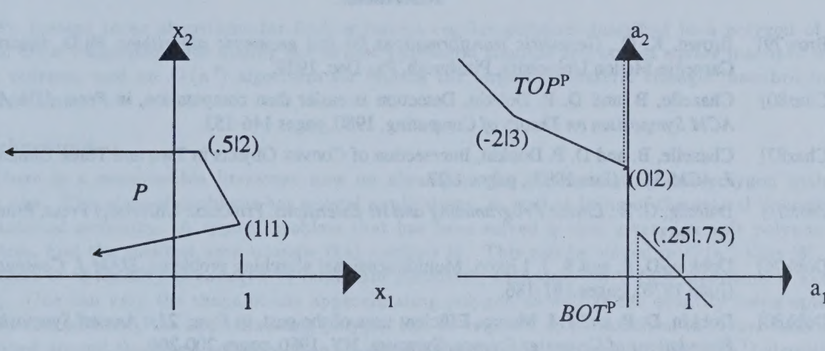


Figure 5.1

5.2. Vertical Hyperplanes

Vertical hyperplanes pose a problem for the dual scheme because they do not have a dual point with finite coordinates. However, for each vertical hyperplane h there is a virtual dual point at infinity. Let (h_n) denote a sequence of non-vertical hyperplanes that converges towards h, such that all hyperplanes h_n have the same $(d-2)$-dimensional point set Q in common (i.e. Q is the intersection of any two hyperplanes h_{n_1} and h_{n_2}).

Let $F_{h_n}(x_1 \, .. \, x_{d-1}) = a_1^n x_1 + \, .. \, + a_{d-1}^n x_{d-1} + a_d^n$. As described in section 3, $TOP^P(a_1^n \, .. \, a_{d-1}^n)$ is obtained as follows. First, one performs a $(d-1)$-dimensional point location in the projection of TOP^P's graph on the hyperplane $J : a_d = 0$ to find the partition $E \subseteq J$ that contains the point $(a_1^n \, .. \, a_{d-1}^n)$. Then, one computes the functional value $F_{D(v(E))}(a_1^n \, .. \, a_{d-1}^n)$, where $v(E)$ is the vertex of P that corresponds to the partition E.

Lemma 5.1: There is an $n_0 \in N$ such that for all $n > n_0$ all duals $D(h_n)$ belong to the same partition $\bar{E} \subseteq J$.

Proof: Because Q is a subset of each hyperplane h_n, each dual point $D(h_n)$ lies on the dual non-vertical straight line $D(Q)$. Clearly, $(D(h_n))$ goes to infinity as n goes to infinity. On the other hand, each partition $E \subseteq J$ is convex, and the number of partitions is finite. From there, the lemma follows. □

In order to check h for intersection with some polyhedron P, one can now proceed similarly as in the case of a non-vertical hyperplane. The partition $\bar{E}$ can be obtained by a point location. Then, one computes the two limits $\lim_{n \to \infty} (TOP^P(a_1^n \, .. \, a_{d-1}^n) - F_{D(v(\bar{E}))}(a_1^n \, .. \, a_{d-1}^n))$ and $\lim_{n \to \infty} (F_{D(v(\bar{E}))}(a_1^n \, .. \, a_{d-1}^n) - BOT^P(a_1^n \, .. \, a_{d-1}^n))$. h intersects P if and only if both limits are greater or equal zero. Moreover, h supports P if and only if at least one of the limits is finite.

6. Conclusions

We showed that in arbitrary, but fixed dimensions, the hyperplane-polyhedron and the polyhedron-polyhedron intersection detection problems can be solved in logarithmic and polylogarithmic time, respectively. For dimensions larger than three, these results appear to be new. There are two reasons why, as of now, these results are of primarily theoretical interest. First, the coefficient which is exponential in d becomes prohibitively high for higher dimensions. Second, the space and preprocessing requirements are not suitable for practical purposes. It is subject to further research to improve these results in order to achieve practical algorithms for intersection detection in higher dimensions. In particular, we suspect that lower bounds for space and preprocessing may be achieved at the expense of slightly higher time bounds for the detection algorithms.

References

[Brow79] Brown, K. Q., *Geometric transformations for fast geometric algorithms,* Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, Pa., Dec. 1979.

[Chaz80] Chazelle, B. and D. P. Dobkin, Detection is easier than computation, in *Proc. 12th Annual ACM Symposium on Theory of Computing*, 1980, pages 146-153.

[Chaz87] Chazelle, B. and D. P. Dobkin, Intersection of Convex Objects in Two and Three Dimensions, *J. ACM* **34**, 1 (Jan. 1987), pages 1-27.

[Dant63] Dantzig, G. B., *Linear Programming and Its Extensions,* Princeton University Press, Princeton, NJ, 1963.

[Dobk76] Dobkin, D. P. and R. J. Lipton, Multidimensional searching problems, *SIAM J. Comput.* **5**, 2 (June 1976), pages 181-186.

[Dobk80] Dobkin, D. P. and J. I. Munro, Efficient uses of the past, in *Proc. 21st Annual Symposium on Foundations of Computer Science*, Syracuse, NY, 1980, pages 200-206.

[Dobk83] Dobkin, D. P. and D. G. Kirkpatrick, Fast detection of polyhedral intersection, *Theoret. Comput. Sci.* **27** (1983), pages 241-253.

[Edel86a] Edelsbrunner, H., L. J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, in *SIAM J. Comput.* **15**, 1986, pages 317-340.

[Edel86b] Edelsbrunner, H., J. O'Rourke, and R. Seidel, Constructing arrangements of lines and hyperplanes with applications, in *SIAM J. Comput.* **15**, 1986, pages 341-363.

[Gunt87] Gunther, O., *Representation of geometric data,* Ph.D. dissertation, University of California, Berkeley, Ca., in preparation, December 1987.

[Lee84] Lee, D. T. and F. P. Preparata, Computational geometry - a survey, *IEEE Trans. on Computers* **C-33**, 12 (Dec. 1984), pages 1072-1101.

[Megi84] Megiddo, N., Linear programming in linear time when the dimension is fixed, *J. ACM* **31**, 1 (Jan. 1984), pages 114-127.

[Newm79] Newman, W. M. and R. F. Sproull, *Principles of Interactive Computer Graphics,* McGraw-Hill, New York, NY, 1979.

[Prep85] Preparata, F. P. and M. I. Shamos, *Computational geometry,* Springer-Verlag, New York, NY, 1985.

[Rock70] Rockafellar, R. T., *Convex Analysis,* Princeton, NJ, 1970.

[Ston86] Stonebraker, M., T. Sellis, and E. Hanson, An analysis of rule indexing implementations in data base systems, in *Proc. of the 1st International Conference on Expert Data Base Systems*, April 1986.

FINDING LARGEST INSCRIBED EQUILATERAL TRIANGLES AND SQUARES

N. ADLAI DE PANO[1], *YAN KE* and *JOSEPH O'ROURKE*

Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218

[1]Now at University of New Orleans

ABSTRACT

We present three algorithms for finding largest regular polygons inscribed in a polygon of n vertices: $O(n^2)$ algorithms for finding the largest equilateral triangle and largest square inscribed in a convex polygon, and an $O(n^3)$ algorithm for finding the largest equilateral triangle inscribed in a general polygon.

1. Background

There is a considerable literature now on algorithms for approximating one polygon with a simpler one. This class of problems has several applications, as well as being of theoretical interest in computational geometry. A typical problem that has been solved is this: given a simple polygon of n vertices, find the smallest area triangle that encloses it. This can be solved in $O(n)$ time [6]. If the criteria of optimality is changed to minimum perimeter, then only an $O(n^3)$ algorithm is available [3]. One can vary the shape of the approximating polygon as well as the quantity being optimized. And in addition, the approximating polygon can be required to be inscribed in rather than circumscribed around the given polygon. For inscribed polygons, the best result is an $O(n^7)$ algorithm for finding the largest area convex polygon inscribed in a polygon [2].

In this paper we examine the problem of finding largest area *regular* convex polygons inscribed in a polygon. In particular, we show that the largest equilateral triangle and largest square inscribed in a *convex* polygon, each may be found in $O(n^2)$ time. Note here that we restrict the input polygon to be convex. Despite this severe restriction, the algorithms are not trivial. Each algorithm is based on a geometric theorem that partially characterizes maximal inscribed regular polygons. Our third result is an $O(n^3)$ algorithm for finding the largest equilateral triangle inscribed in a general (non-convex) polygon. We did not attempt seriously to solve the inscribed square problem for general polygons, since the triangle algorithm is already rather complex.

Restriction to regular approximating polygons was first studied in [4], wherein it was shown that the smallest circumscribed equilateral triangle and square can each be found in $O(n)$ time. Our results show that the corresponding inscribing problems are more difficult.

The three algorithms are described in the next three sections, starting with the simplest and finishing with the most complex. Some details will have to be omitted due to space limitations. Conclusions are drawn in the final section of the paper.

2. Largest Equilateral Triangle Inscribed in a Convex Polygon

Let P be the given convex polygon of n vertices. It is easy to see that if the inscribed triangle T is not restricted to be equilateral, then all of its vertices must coincide with vertices of P [1]. This is not obvious for the more constrained case, and in fact is not true in general. But we can prove that at least one vertex of T must coincide with a vertex of P:

Theorem 1: A maximum area equilateral triangle inscribed in a convex polygon P must have at least one corner coincident with a vertex of P.

Proof: This theorem was first stated and proved in [3]. The proof there showed that if T had no corner at a vertex of P, then the function describing T's area in terms of its orientation could not be at a maximum. The algebraic details of this proof are formidable. Here we offer a simpler geometric proof.

Suppose T has no corners at the vertices of P. Let e_a, e_b and e_c be the three edges of P that contain the three corners a, b and c of T respectively. Let r_a be the ray perpendicular to e_a at a, directed towards the interior of P, and similarly let r_b be the perpendicular to e_b at b. Let p be the intersection of r_a and r_b. A slight rigid rotation of p about P clockwise (cw) or counterclockwise (ccw) will detach both a and b from e_a and e_b, leaving them strictly inside P. Depending on the angle formed by e_c and pc, a slight rotation of T either cw or ccw will pull c to be strictly inside P. Since P is convex, T will be strictly inside P after an appropriate rotation of T around p. Then the area of T can be increased by an expansion of T which contradicts the assumption. □

Theorem 1 permits our algorithm to have the following high-level structure:

```
Algorithm $T_1$
    for each vertex $v$ of $P$ do
        Find the largest equilateral triangle with one corner at $v$.
        Save max.
    return max.
End of $T_1$.
```

So we now concentrate on the step inside the loop, which is repeated n times.

Let the triangle T have corners a, b, and c, and let corner a be anchored at vertex 0 of P, with the remaining vertices of P numbered to increase clockwise from 0. The plan is to move corner b from 0, along edge (0,1), then along edge (1,2), and so on, adjusting T accordingly, ignoring whether or not T stays totally inside P. Fig. 1 shows two snapshots of this process. If the path of corner c is traced out as b traverses the boundary of P, it traces out an exact copy of P, rotated by 60° about vertex 0. This claim can be seen as follows. Let $|xy|$ denote the length of the segment from x to y. In Fig. 1, $|ab| = |ac|$ and $|ab'| = |ac'|$. Moreover $\angle bab' = \angle cac'$. Therefore, triangle (b, a, b') is congruent to triangle (c, a, c'), implying that $|bb'| = |cc'|$ and $\angle abb' = \angle acc'$. Thus the movement of b is copied exactly by the movement of c.

Let C be the rotated polygon traced out by c. It has the property that if $c \in C$, then for the corresponding b, $b \in P$.

Now we want both $b \in P$ and $c \in P$. This can be achieved by requiring that $c \in P \cap C$, since $c \in C \rightarrow b \in P$. Thus the largest triangle inscribed in P is determined by the longest segment ac to a point in $P \cap C$. See Fig. 2. This longest chord is either achieved with c at a vertex of P or C, or c at the intersection of edges of P and C. In the former case (as in Fig. 2), T shares two vertices with P, but in the latter, it shares just a.

Now the algorithm is immediate. Rotate P by 60° to obtain C. Intersect P and C in linear time [7]. Then find the longest ac chord in $P \cap C$ by a linear scan around its boundary. Thus, in $O(n)$ time, the largest equilateral triangle with one corner anchored is found. Repeating this n times, once per choice of anchoring vertex, yields an $O(n^2)$ algorithm for the entire problem. We have tried to obtain a subquadratic algorithm without success.

3. Largest Square Inscribed in a Convex Polygon

The algorithm for finding the largest square inscribed in a convex polygon is more complicated because Theorem 1 does not hold. Rather we have this characterization:

Theorem 2: A maximum area square inscribed in a convex polygon P must satisfy at least one of the following two conditions:

(1) At least one corner is coincident with a vertex of P, or

(2) All four corners lie on the interior of edges of P.

Proof: Suppose neither condition holds. Then at most three corners of S lie on the interior of edges of P. If only one or two corners touch, then clearly S can be enlarged. If exactly three touch, then the right triangle formed by these three corners can be increased in area by the same technique as used in Theorem 1. In all cases, the assumed maximality of S is contradicted. $\square$

Our algorithm handles each possibility separately. The first case is similar (but a bit more complicated) than that for the equilateral triangle. The second case is quite different and requires a new technique. Both cases can be explored in $O(n^2)$ time. We now proceed to the details.

3.1. One Corner Coincident

Let the square S have corners a, b, c and d labeled cw, and let corner a be anchored at vertex 0 of P just as in Section 2. Move corner c of S around the entire boundary of P from 0, along (0,1), then (1,2), and so on. Let B and D be the polygons traced out by corners b and d during this process. Then we claim that B is a copy of P reduced in size by a factor of $1/\sqrt{2}$, and rotated 45 ° counterclockwise about a, and D is a similarly reduced copy of P, but rotated 45 ° cw. The proof is straightforward and omitted.

Now we want $b \in P$, $c \in P$ and $d \in P$. This can be achieved by choosing $b \in P \cap B$ and $d \in P$. But the choices for b and d must correspond to the same square. Let D' be the polygon $P \cap D$ rotated ccw by 90 ° about a. This rotation maps d onto its corresponding b. If we now choose $b \in Q = (P \cap B) \cap D'$, all of $b \in P$, $c \in P$ and $d \in P$ are achieved simultaneously. See Fig. 3.

The algorithm for this case is now apparent. Compute B, D, $P \cap B$, $P \cap D$, D', and Q, in that order, each in linear time. Find the longest ab chord in Q by a linear scan of its boundary. Repeating this procedure for each of the n anchoring choice yields an $O(n^2)$ time algorithm for this case.

3.2. Four Corners on Edges

Now consider the case when all four corners of S lie on the interior of edges of P. At a high level, our algorithm will be:

```
Algorithm $S_1$
    for each edge $e$ of $P$ do
        Find the largest equilateral triangle with one corner on $e$.
        Save max.
    return max.
End of $S_1$.
```

In order to achieve $O(n^2)$, the step inside the loop must be accomplished in $O(n)$ time. That step will have the following structure, for $e = e_0$. Let the edges of P be labeled e_0, e_1, ..., e_{n-1} cw.

```
Algorithm $S_2$
    for $i = 1$ to $n-1$ do
        Find the largest square with corner $a$ on $e_0$ and corner $c$ on $e_i$.
        Save max.
    return max.
End of $S_2$.
```

The key to the efficient implementation of the Algorithm S_2 is the geometric structure revealed in the following lemma.

Lemma 1: Let $B(e_i, e_j)$ be the positions assumed by corner b of the square S when corner $a \in e_i$ and corner $c \in e_j$; similarly define $D(e_i, e_j)$ for corner d. Then $B(e_i, e_j)$ and $D(e_i, e_j)$ are parallelograms, whose four vertices correspond to the four extreme positions of a and c at the endpoints of e_i and e_j.

We will not prove this formally, but rather just depend on Fig. 4 for justification. Every point $b \in B(e_i, e_j)$ maps to points $a \in e_i$ and $c \in e_j$ in the following natural manner. Let z be the corner of $B(e_i, e_j)$ representing the upper endpoints of e_i and e_j, as in Fig. 4. Then if a is a distance d_i along e_i from e_i's upper endpoint, and c is a distance d_j along e_j, then the corresponding b point has oblique coordinates $(d_i/\sqrt{2}, d_j/\sqrt{2})$ inside $B(e_i, e_j)$, measured parallel to the sides of the parallelogram from an origin at z. The same is true for $D(e_i, e_j)$.

Because we are only interested in the squares all of whose four corners touch P, we need only examine points b on the boundary of $P \cap B(e_i, e_j)$, and similarly for $D(e_i, e_j)$. As in Section 3.1, we want to transform the d points so they coincide with their b counterparts. Let D' be the appropriate transformation of $P \cap D(e_i, e_j)$ that maps $D(e_i, e_j)$ to $B(e_i, e_j)$. This transformation is a reflection and then a linear "skewing"[1], and so can be performed in linear time. If we now choose $b \in Q = (P \cap B(e_i, e_j)) \cap D'$, then we will have $a \in e_i$, $b \in e_j$, and $b \in P$ and $d \in P$. If we choose b to be at intersection of the boundary of $P \cap B(e_i, e_j)$ and the boundary of D', then in addition both b and d will also lie on edges of P. This then yields an algorithm for one (e_i, e_j) pair.

The complexity of this algorithm is evidently proportional to the size of $P \cap B(e_i, e_j)$ and $P \cap D(e_i, e_j)$, that is, the number of edges of P that intersect these parallelograms. As this number could be $\Omega(n)$ for any particular parallelogram, our task now is to show that the sum over all parallelograms for (e_0, e_i), with e_0 fixed and e_i varying, is nevertheless only $O(n)$. This will then yield an $O(n^2)$ algorithm overall.

The key idea is stated in the following lemma without proof.

Lemma 2: Let 0 and 1 be the labels of the two endpoints of e_0. Define P_0 to be rotated 45° cw around 0 and scaled by a factor of $1/\sqrt{2}$, and define P_1 similarly except rotated around 1. If the corresponding vertices of P_0 and P_1 are connected, the resulting edges are precisely all the edges in the parallelograms $B(e_0, e_1)$, $B(e_0, e_2)$, ..., $B(e_0, e_{n-1})$. The same holds true for $D(e_0, e_i)$ if the rotation is 45° ccw.

Again we depend on Fig. 5 for justification. Now note that, due to the convexity of P_0 and P_1, the parallelograms may be partitioned into two "belts" such that none of the parallelograms overlap within one belt. But this then implies that the number of edges that intersect each parallelogram, summed over all parallelograms, is $O(n)$. This establishes that all relevant squares with corner a on e_0 can be found in $O(n)$ (Algorithm S_2), and therefore yields an $O(n^2)$ algorithm overall.

4. Largest Equilateral Triangle Inscribed in a General Polygon

If the surrounding polygon is not convex, then Theorem 1 may fail to hold even for equilateral triangles. In this case we have this characterization:

Theorem 3: A maximum area equilateral triangle T inscribed in a general polygon P must satisfy at least one of the following two conditions:

(1) At least one corner is coincident with a convex vertex of P, or

(2) At least one edge is supported in its interior by at least one reflex vertex of P.

Proof: Clearly two edges of T must be blocked from equilaterally expanding if T is to be maximum. If neither condition of the theorem holds, this blockage could only occur with either all three corners of T or edges, or a corner hitting a reflex vertex. Suppose neither condition holds. If three corners of T lie on the interior of edges of P, then T can not have maximum area by Theorem 1. If there is a corner of T that is coincident with a reflex vertex of P, then we can construct a straight line through the vertex that locally separates T and P. The straight line can be treated as an edge of P. Then again by Theorem 1, T can not be maximum. □

The non-convexity of P lead to quite a number of complexities whose resolution requires $O(n^3)$ time. Our algorithm handles each possibility separately. We now proceed to the details.

4.1. One Corner Coincident

Let a, b and c be the three corners of T labeled cw, and let corner a be anchored at vertex 0 of P just as in Section 2. It is not hard to see that the two corners b and c must lie inside the visibility polygon of a. Thus we first construct the visibility polygon P_a of a in $O(n)$ time [5]. All the vertices of P_a, except a are sorted angularly about a. Now we rotate P_a around a by 60° ccw and call this new polygon P_a'. By the argument in Section 2, if $b \in P_a \cap P_a'$, then $c \in P_a$. The intersection region $P_a \cap P_a'$ can be computed in $O(n)$ time. However, since P is not convex, T may still intersect the exterior of P even when both b and c are in P. We now describe how to handle these intersections.

---

[1] The angle of $B(e_i, e_j)$ at z is $90° + \theta$, where θ is the angle formed by the lines containing e_i and e_j, and the angle of $D(e_i, e_j)$ at z' is $90° - \theta$.

Consider the cone region obtained by extending into rays the two edges of $P_a \cap P_a'$ adjacent to a. Let $v_1 < v_2 < \ldots < v_p$ be the vertices of P_a that are inside this cone region in ccw order Similarly define $v_1' < v_2' < \cdots < v_q'$ for P_a'. We can then merge these two lists into one sorted list L_a in $O(n)$ time. By connecting a to each vertex in the list L_a, we partition $P_a \cap P_a'$ into a collection of triangles $t_1, t_2, \ldots, t_a$, each incident to vertex a. Let $V(b)$ be the set of vertices of P_a between b and c in the angular order. Clearly $V(b)$ remains constant for all b inside a single triangle t_i. The proof of the following lemma is straightforward and omitted.

Lemma 3: T is inside P iff all vertices of $V(b)$ are on the opposite of bc from a.

For an arbitrary $v \in V(b)$, let $R(v)$ be the circle sector region defined as follows. Let C be a circle passing through a and c, such that the circular arc from a to c subtends $120°$ at the center of the circle, and such that this arc is to the same side of ac as b. $R(v)$ is the region bounded by the chord ac of C, and the arc of C from a to c.

Lemma 4: T is inside P iff b is inside both t_i and $\bigcap_{v \in V(b)} R(v)$, where $i \in \{1, \ldots, a\}$.

Proof: *[if]*. Suppose T is not completely inside P. Then by Lemma 3, there is a vertex $v \in V(b)$ such that v is on the same side of bc as a. But then $\angle abv < 60°$ because if b were on the arc of $R(v)$, the angle would be exactly $60°$. This implies b is not inside $R(v)$, a contradiction.

[only if]. Let v be an arbitrary vertex in $V(b)$. If T is inside P, then by Lemma 3, v must be on opposite side of bc from a. If follows that $\angle abv \geq 60°$ Thus $b \in R(v)$. Since v is chosen arbitrarily, we have $b \in \bigcap_{v \in V(b)} R(v)$. □

The above lemma suggests the following algorithm for a convex vertex 0 of P.

```
Algorithm $T_2$
    for $i = 1$ to $a$ do
        For $b \in t_i$, compute $V(b)$ and $\bigcap_{v \in V(b)} R(v)$.
        Find the largest $T$ with $b$ inside $t_i$ and $\bigcap_{v \in V(b)} R(v)$.
        Save max.
    return max.
End of $T_2$.
```

Since the number of iterations in the above algorithm is $O(n)$, if we can achieve $O(n)$ per iteration, we will get an $O(n^2)$ algorithm for each convex vertex of P. The intersection $\bigcap_{v \in V(b)} R(v)$ can be computed by a straightforward $O(n \log n)$ incremental algorithm, which inserts each $R(v)$ into the partially-constructed intersection one at a time. With a little bit of care, we can achieve a better result. Let $v_1, v_2, \ldots, v_b$ be the vertices in $V(b)$ in the sorted order. Suppose we have obtained $R_i = \bigcap_{j=1}^{i} R(v_i)$ and are about to insert $R(v_{i+1})$. Clearly the boundary of R_i is formed by the line segment av_1 and a sequence of arcs. A simple induction on i shows that each $R(v_j)$ contributes at most one arc for R_i, where $j \in \{1, \ldots, i\}$. When $R(v_{i+1})$ arrives, we traverse the boundary of R_i from a clockwise until we find an arc $\hat{a}$ intersecting the arc of $R(v_{i+1})$ at a point p different from a. See Fig. 6. Then we split the boundary of R_i at p and replace the part from a to p by the arc of $R(v_{i+1})$. Since each arc is traversed at most twice, the final intersection $R_b = \bigcap_{v \in V(b)} R(v)$ can be found in $O(n)$ time. Since the size of $t_i \cap R_b$ is $O(n)$, the longest chord ab can be found by a linear scan around its boundary. Therefore algorithm T_2 runs in $O(n^2)$ time, and repeating it for each of $O(n)$ convex vertices of P results in an $O(n^3)$ algorithm covering the first condition of Theorem 3. We now turn to the second condition.

4.2. One Edge Supported by a Reflex Vertex

Now we consider the case when there is an edge of T that is supported in its interior by a reflex vertex of P. At a high level, our algorithm will be:

```
Algorithm $T_3$
    for each reflex vertex $v$ of $P$ do
        Find the largest equilateral triangle with one edge supported by $v$.
        Save max.
    return max.
End of $T_3$.
```

In order to achieve $O(n^3)$, the step inside the loop must be accomplished in $O(n^2)$ time. Fig. 7 shows all the configurations a maximum T can achieve when one of its edges is supported by v. Again the corners a, b and c must lie inside the visibility polygon of v. So we first spend $O(n)$ time to compute the visibility polygon P_v of v. Then all vertices of P_v are sorted angularly about v cw. Consider a chord h of P that passes through the vertex v. A chord is a maximal line segment inside P, maximal in the sense that it connot be extended without intersecting the exterior of P. We say h is *critical* if either of its endpoints is coincident with a vertex of P_v. It is clear that the number of critical chords is $O(n)$. Two critical chords h_1 and h_2, adjacent in a rotation about v, form a pair of triangles apexed at and sharing v, with bases on the boundary of P_v. Every chord h through v that lies between h_1 and h_2, has its left endpoint p_l on the same edge e_l, and its right endpoint p_r on the same edge e_r. The algorithm for each reflex vertex of P has the following outline:

```
Algorithm $T_4$.
    for each pair of adjacent critical chords $h_1$ and $h_2$ do
        Find the largest $T$ whose edge $ab$ lies between $h_1$ and $h_2$.
        Save max.
    return max.
End of $T_4$.
```

In order to achieve $O(n^2)$, the step inside the loop must be accomplished in $O(n)$ time. We next proceed to a detailed discussion of this step.

Let T have its corner a coincident with p_r and b with p_l. As T rotates about v, the trace of the third corner c is an algebraic curve with constant degree. We can split this curve into a constant number of pieces such that each piece is angularly monotone with respect to v. The pair of triangles formed by h_1 and h_2 are also partitioned accordingly. If we can show each piece can be handled in $O(n)$ time, then we are finished. So wlog, let the trace of c is monotone with respect to v. We denote this curve by ω and two end points α_l and α_r. Let $V(\omega)$ be the set of vertices of P_v that are between α_l and α_r in angular order. For each vertex $v' \in V(\omega)$, there is a chord h that satisfies the following three conditions:

(1) h passes through v;

(2) h is between h_1 and h_2;

(3) If the T edge $ab = h$, then corner c is collinear with v' and v;

All these chords will partition the triangle pair formed by h_1 and h_2 into a collection of finer triangle pairs. In the remainder of this section, we first describe an $O(n^2)$ algorithm for this particular triangle pair formed by h_1 and h_2. It has the following high-level structure:

```
Algorithm $T_5$
    for each finer triangle pair in the collection do
        Find the largest $T$ whose edge $ab$ is oriented within the finer triangle pair.
        Save max.
    return max.
End of $T_5$.
```

where each step inside the loop takes $O(n)$ time. Then we show how to improve this algorithm to run in $O(n)$ time as required for the total $O(n^3)$ time complexity.

Let (t_1,t_1'), (t_2,t_2'), ..., (t_s,t_s') be the finer triangle pairs for the two adjacent critical chords h_1 and h_2, and let t_i be to the left of t_i' for $i \in \{1,...,s\}$. Let h be a chord which passes through v and is inside one finer triangle pair (t_i, t_i'). If we take $h = ab$ of T, then the corner c is on the curve ω. Let $V(t_i) \subseteq V(\omega)$ containing vertices of smaller index c. Define $V(t_i')$ symmetrically. Then it is clear that no matter how h varies within (t_i, t_i'), both $V(t_i)$ and $V(t_i')$ remain constant. We omit the simple proof of the following lemma.

Lemma 5: T contains no vertices of P iff all vertices in $V(t_i)$ and $V(t_i')$ are on opposite sides of bc and ac from v respectively.

For each $v_1 \in V(t_i)$, let $R(v_1)$ be the region bound by the segment vv_1, and the arc of a circle C, where C passes through v and v_1, its circular arc from v to v_1 is to the same side of vv_1 as e_l and subtends 120 ° at its center. Similarly define $R(v_1)$ for each $v_1 \in V(t_i')$. The following Lemma follows directly from Lemma 4 and 5.

Lemma 6: T contains no vertices of P iff a is contained in $Q(t_i) = \bigcap_{v \in V(t_i)} R(v)$ and b is contained in $Q(t_i') = \bigcap_{v \in V(t_i')} R(v)$.

Again both regions $Q(t_i)$ and $Q(t_i')$ can be found in $O(n)$ time. Then we compute $Q_t(t_i) = t_i \cap Q(t_i)$, also in $O(n)$ time. Similarly for $Q_t(t_i') = t_i' \cap Q(t_i')$. By a linear scan, we can find all positions of h such that at least one endpoint is coincident with a vertex of $Q_t(t_i)$ or $Q_t(t_i')$. The chords at these positions further partition $Q_t(t_i)$ and $Q_t(t_i')$ into a collection of *double wedges*, each of which is bounded by outward convex arcs or straight lines at both sides. Let w_l and w_r form a double wedge, with w_l to the left of w_r.

Case 1. Both w_l and w_r are bounded by straight lines.

Let the edge ab of T be supported by v and inside w_l and w_r. If c is inside P when ab achieves maximum, we have the situation shown in Fig. 7(a). Otherwise, we either have Fig. 7(b) or Fig. 7(c). In all cases, since the edges of P that are possibly touched by a, b and c are known, we can find the locally largest T in constant time.

Case 2. Both w_l and w_r are bounded by arcs.

Again let the edge ab be supported by v and inside w_l and w_r. We have three ways of achieving the locally largest T as shown in Fig. 7(d)(e)(f). Since the edge of P that is possibly touched by c, and the two vertices of P that possibly support the edges bc and ca are all known, we can find the local maximum in constant time.

Case 3. One wedge is bounded by arc and the other is by a straight line.

Wlog, let w_l (w_r) be bounded by straight line (arc). This gives three possibilities of achieving the local maximum. See Fig. 7(g)(h)(i). Again, all the relevant edges and vertices are fixed, so we need only constant time to handle it. The symmetric situation is when w_l (w_r) is bounded by arc (straight line). The corresponding possible maximums are shown in Fig. 7(j)(k)(l).

We have shown that each double wedge can be handled in constant time. Since there can be $O(n)$ such double wedges for each finer triangle pair, the time complexity of Algorithm T_4 is $O(n^2)$.

We now discuss the improvement. Let $Q_e(t_i)$ and $Q_e(t_i')$ be the set of edges of $Q(t_i)$ and $Q(t_i')$ respectively, for $i \in \{1,...,s\}$. Notice an edge in $Q_e(t_i')$ or $Q_e(t_i')$ can be an arc.

Lemma 7: The size of $\bigcup_{i=1}^{s} Q_e(t_i)$ is $O(s)$. The same is true for $\bigcup_{i=1}^{s} Q_e(t_i')$.

Proof: Define $Q_i = \bigcup_{j=1}^{i} Q_e(t_i)$. Now consider the condition of $Q_e(t_{i+1})$. By the incremental algorithm introduced previously, the number of new edges introduced by $Q_e(t_{i+1})$ is at most three. Thus the size of $Q_s = \bigcup_{j=1}^{s} Q_e(t_i)$ is $O(s)$. The same argument holds for $\bigcup_{i=1}^{s} Q_e(t_i')$. □

Let $E(t_i)$ be the set of edges of $Q(t_i)$ that are contained inside t_i. Similarly define $E(t_i')$. By the above algorithm, $E(t_i)$ and $E(t_i')$ are all that we need for the finer triangle pair (t_i, t_i').

Lemma 8: The size of $\bigcup_{i=1}^{s} E(t_i)$ is $O(s)$. The same is true for $\bigcup_{i=1}^{s} E(t_i')$.

Proof: Consider one triangle t_i with two edges e_1 and e_2 adjacent to v. The set $E(t_i)$ of edges forms a convex chain. There are at most two edges in this chain that can touch e_1 or e_2. Clearly all edges in $E(t_i)$, except those two, are in $Q_e(t_i)$ and distinct from any other edges in $Q_e(t_j)$, for all $j \in \{1, ..., i-1, i+1, ..., s\}$. This implies the correctness of the lemma. □

By using the incremental algorithm introduced in the previous section, we can compute $Q(t_i)$ from $Q(t_{i-1})$ in amortized $O(1)$ time. To compute $Q(t_i')$ from $Q(t_{i-1}')$, we simply reverse the incremental algorithm with the same time complexity. It follows that algorithm T_5 runs in $O(n)$ time. This gives an overall $O(n^3)$ time algorithm for computing the largest T inside P.

5. Conclusion

We have presented two $O(n^2)$ algorithms for computing the largest equilateral triangle and square inscribed inside a given convex polygon P. In general, the maximum is not unique. An interesting question is: what is largest number of maximal equilateral triangles or squares that are admitted by a given convex polygon P? A trivial lower bound $\Omega(n)$ can be achieved when P is regular. But the upper bound is unknown.

We also give an $O(n^3)$ algorithm for computing the largest equilateral triangle inside a simple polygon. The corresponding largest square problem seems more difficult. We do not know how to solve the case when four of its corners are on the interior of the edges of P in $O(n^3)$ time.

It is not hard to see that the techniques presented in this paper can also be used to solve the following problem: Given a convex polygon P and a triangle T (quadrilateral Q), find the largest inscribed triangle (quadrilateral) that is similar to T (Q). The corresponding triangle problem in a simple polygon can also be solved.

REFERENCES

[1] J. Boyce, D. Dobkin, R. Drysdale, L. Guibas, "Finding extremal polygons," *Proc. 14th Ann. Symp. on Theory of Comput.*, 282-289 (1982).

[2] J. Chang and C.Yap, "A polynomial solution for the potato-peeling problem," *Discrete and Computational Geometry* **1**, 155-182 (1986).

[3] A. De Pano, "Polygon approximation with optimized polygonal enclosures: applications and algorithms," PhD Thesis, Johns Hopkins University, June 1987.

[4] A. De Pano and A. Aggarwal, "Finding restricted k-envelopes for convex polygons," *Proc. 22nd Allerton Conf. on Comm. Cont. and Comput.*, 81-90 (1984).

[5] D.T. Lee and A.K. Lin, "Computing the visibility polygon from an edge," *Computer Vision, Graphics and Image Processing* 34, 1986, 1-19.

[6] J. O'Rourke, A. Aggarwal, S. Maddila, M. Baldwin, "An optimal algorithm for finding minimal enclosing triangles," *J. of Algorithms* **7**, 258-269 (1986).

[7] J. O'Rourke, C.-B. Chien, T. Olson, D. Naddor, "A new linear algorithm for intersecting convex polygons," *Comput. Graphics and Image Proc.* **19**, 384-391 (1982).

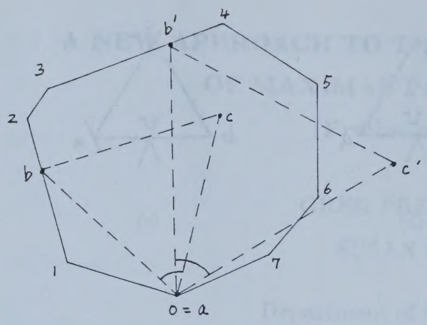

b'
4
3
5
2
c
b
c'
6
1
7
0 = a

Figure 1.

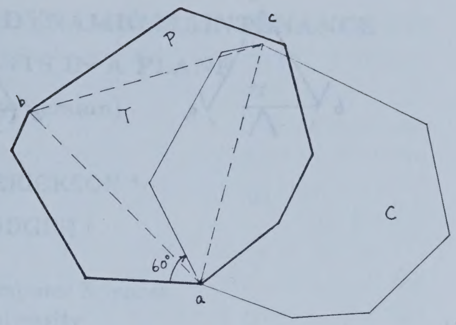


Figure 2.

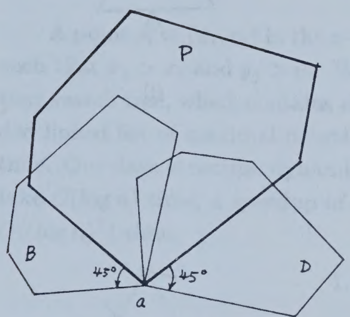


Figure 3.

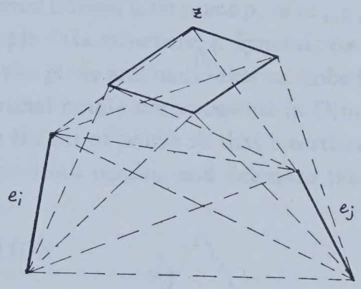


Figure 4.

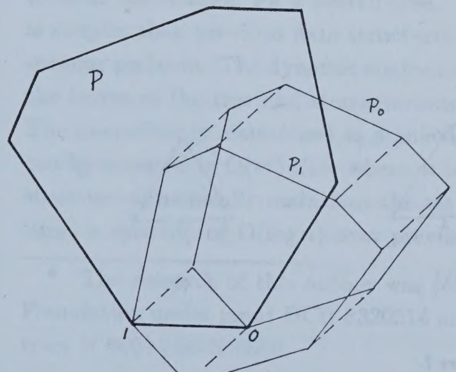


Figure 5.

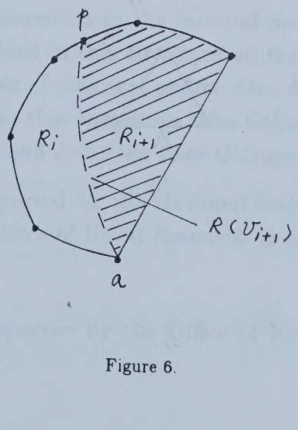
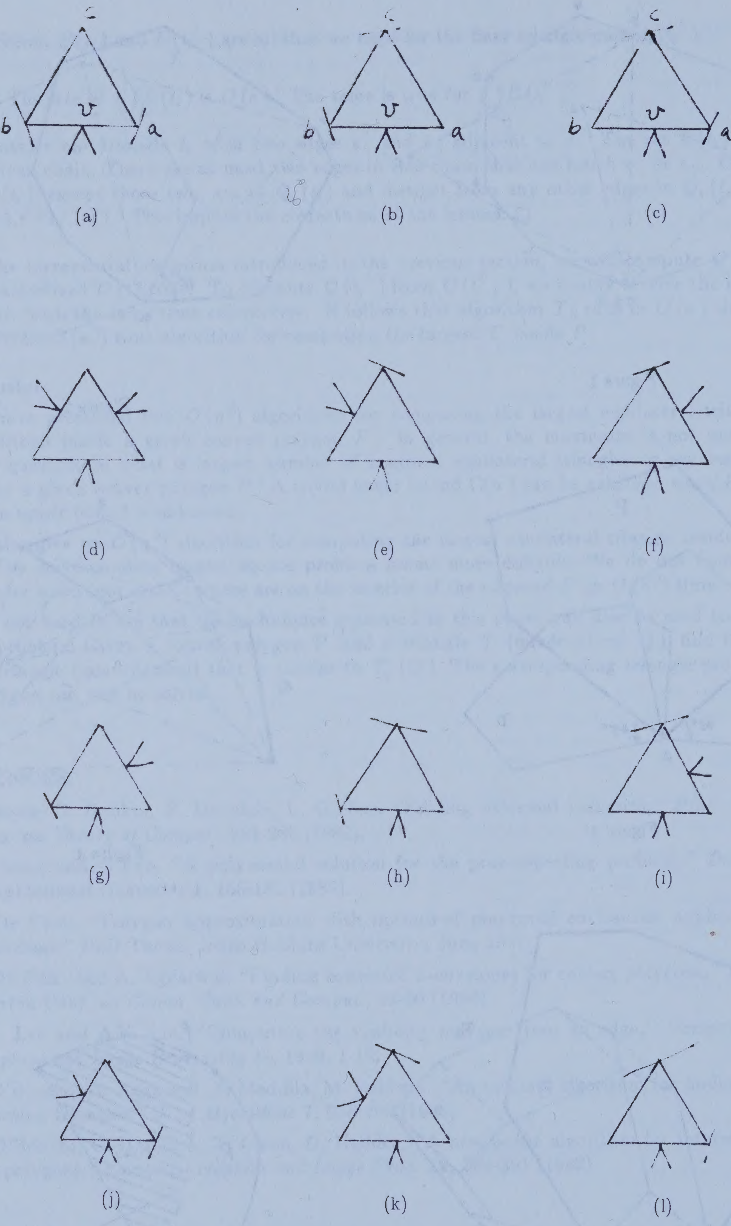

p
R_i
R_{i+1}
$R(v_{i+1})$
a

Figure 6.


c
v
b
a
c
v
b
a
c
v
b
a

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 7.

A NEW APPROACH TO THE DYNAMIC MAINTENANCE OF MAXIMAL POINTS IN A PLANE

(Preliminary Version)

GREG FREDERICKSON *
SUSAN RODGER †

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

ABSTRACT

A point $p_i = (x_i, y_i)$ in the x–y plane is *maximal* if there is no point $p_j = (x_j, y_j)$ such that $x_j > x_i$ and $y_j > y_i$. We present a simple data structure, a dynamic contour search tree, which contains all the points in the plane and maintains an embedded linked list of maximal points so that m maximal points are accessible in $O(m)$ time. Our data structure dynamically maintains the set of points so that insertions take $O(\log n)$ time, a speedup of $O(\log n)$ over previous results, and deletions take $O((\log n)^2)$ time.

1. INTRODUCTION

Given a set S of points in the x–y plane, $p_i = (x_i, y_i) \in S$ is a maximal point if there is no $p_j = (x_j, y_j) \in S$ such that $p_i < p_j$ $(p_i < p_j \iff x_i < x_j$ and $y_i < y_j)$. The set of maximal points of S form what we call an *m-contour.* In this paper we present a dynamic contour search tree, a data structure to represent contour information for a search tree. This data structure is quite natural, and is simpler than previous data structures for solving the dynamic version of the m-contour problem. The dynamic contour search tree stores the points of the set S in the leaves of the tree and stores various search information in the internal nodes. The m-contour is maintained as a linked list embedded in this structure, so that it can be accessed in $O(m)$ time where m is the number of maximal points. Our data structure dynamically maintains the set of points so that insertions take $O(\log n)$ time, a speedup of $O(\log n)$ over previous results, and deletions take $O((\log n)^2)$

---

* The research of this author was partially supported by the National Science Foundation under grant DCR-8320214 and by the Office of Naval Research on contract N 00014-86-K-0689.

† The research of this author was partially supported by the Office of Naval Research on contract N 00014-86-K-0689.

time. In addition, the query of whether a new point lies inside or outside of the m-contour can be determined in O($\log n$) time.

For a static set of n points, Kung et al. [1] have shown that the m-contour can be computed in O($n \log n$) time. One method for constructing the m-contour proceeds by examining all the points, one at a time, and building the m-contour from the points examined thus far. If a point is not on the m-contour then the point is discarded.

In the dynamic version of this problem, those points in the set S which are not maximal points must be saved, because deleting a point on the m-contour can result in a point which was not maximal becoming maximal. Overmars and van Leeuwen [2] use a balanced binary search tree in which the points from S are stored in the leaves of the tree and each internal node has a concatenable queue associated with it. The concatenable queue at the root contains all the maximal points and the other concatenable queues contain the remaining points. In the data structure of Overmars and van Leeuwen, the m-contour can be listed in O(m) time, and the query of whether a new point lies inside or outside of the m-contour can be determined in O($\log n$) time. However, Overmars and van Leeuwen's insertions and deletions consist of tearing apart concatenable queues and rebuilding them at each internal node along a path, causing insertions and deletions to take O($(\log n)^2$) time. With our data structure, insertions and deletions consist of updating O(1) search information at each internal node along a path, causing insertions to take O($\log n$) time.

Willard and Lueker [4] have shown how to perform range queries in k dimensions on trees of bounded balance with worst case times of O($(\log n)^{k-1}$) for update and query operations. Applying their data structure to the maximal point problem in two dimensions results in efficient worst case update and query times of O($\log n$). However, since their data structure is designed specifically for range queries, it has no scheme for representing the m-contour. There appears to be no efficient way to list out the m-contour.

The dynamic contour search tree must be balanced in order to get the O($\log n$) search time. We will use the balancing scheme of the red-black binary search trees of Tarjan [3] since they use the minimum number of rotations to keep the tree balanced. Each rotation in the dynamic contour search tree requires O($\log n$) time to update the search information at the three nodes possibly affected by the rotation. Thus it is crucial to use the red-black binary search trees which have O(1) rotations per update in order to achieve the O($\log n$) time for insertions.

2. DEFINITIONS AND SEARCHING OPERATIONS

The m-contour of a set of points S in the plane is composed of the maximal points of S. These maximal points form a 'staircase' in which all the remaining points of S lie to the left and below the staircase. An example set of points, and the staircase for them, are given in Figure 1a. The *inside region* of the m-contour is the region dominated by the m-contour. That is, a point p_i is inside the m-contour if and only if there is a maximal point p_j such that $p_i < p_j$. The *outside region* of the m-contour is the region that is not dominated by the m-contour.

The data structure we propose to dynamically maintain the m-contour is a balanced binary tree with all the points of the set S sorted by x-coordinate and stored in the leaves of the tree. Each leaf node contains three fields: *x*, *y* and *transfer*. The internal nodes of the tree are used to search for points and to represent the m-contour. Each internal node v contains six fields: *right*, *left*, *max_x*, *max_y*, *next_down* and *next_in_contour*. *Right(v)* is the right child of internal node v and similarly *left(v)* is the left child of v. *Max_x(v)* points to the leaf containing the point with the largest x-coordinate of all the points in v's subtree. We use *max_x(v)* in the search for a point via its x-coordinate. *Max_y(v)* points to the leaf containing the point with the largest y-coordinate of all the points in v's subtree. We use *max_y(v)* when updating certain information after an insertion or deletion. *Next_down(v)* of node v points to the closest descendant node w that has a *max_y(right(w))* = *max_y(right(v))*. If no such w exists, then *next_down(v)* points to the leaf node w that max_y(right(v)) points to. We use *next_down(v)* in sections 3 and 4 to reset *transfer* pointers in O(1) time.

Next_in_contour(v) for internal node v contains a pointer to the descendant leaf in the tree which represents the nearest point that is to the left and above of *max_y(right(v))* in the plane. If *max_y(right(v))* points to the leaf for $p_i = (x_i, y_i)$, then *next_in_contour(v)* will point to some point $p_j = (x_j, y_j)$ which has the largest x-coordinate x_j such that $x_j < x_i$ and $y_j > y_i$. It is possible for several internal nodes to have the same *max_y(right(v))* values. Of these nodes, only the node v closest to the root will have its *next_in_contour(v)* pointer set to the next contour point; the others will set their *next_in_contour* pointers to $NULL$. *Transfer(w)*, where w is a leaf containing the point $p_i = (x_i, y_i)$, is a pointer to the highest ancestor v that has a *y(max_y(right(v)))* value equal to y_i. *Transfer(w)* together with the *next_down* pointers of all the nodes v whose *max_y(right(v))* $= p_i$ form a circularly linked list in the tree.

A dynamic contour search tree for the set of points in Figure 1a is given in Figure 1b. Downward arrows are *next_in_contour* pointers, upward arrows are *transfer* pointers, and a label beside an interior node v is the *max_y(right(v))* value. *Next_down* pointers are not shown. Figure 2 shows the circularly linked list of the *transfer* pointer for the point p_i and the *next_down* pointers of the nodes v whose *max_y(right(v))* $= p_i$.

The m-contour can easily be produced from this data structure by traversing a linked list consisting of alternating *next_in_contour* and *transfer* pointers. Note that the rightmost point on the m-contour will always be a point $p_i = (x_i, y_i)$ with the largest x-coordinate. Starting from p_i, we can trace through the m-contour that is represented in the tree by alternately following *transfer* and *next_in_contour* pointers. This list is emboldened in Figure 1b.

Theorem 2.1 *The points on the m-contour can be listed using the dynamic contour search tree in O(m) time where m is the number of maximal points.*

Proof: The rightmost leaf w in the search tree contains the rightmost point $p_i = (x_i, y_i)$ on the m-contour since the leaves are sorted by x-coordinates and x_i is the largest x-value of any point. *Transfer(w)* points to its highest ancestor v that has a *y(max_y(right(v)))* value of y_i. No point in v's right subtree has a larger y-value. To find the next maximal point, examine *next_in_contour(v)*. It points to the leaf u containing the point $p_k = (x_k, y_k)$ such that $x_k = (\mathit{max}\ x_j \quad \mathit{such\ that}\ x_j < x_i \quad \text{and} \quad y_j > y_i\ )$. Thus by starting with the rightmost leaf and tracing the *transfer* and *next_in_contour* pointers, the m-contour can be listed in O(m) time. □.

An arbitrary point $p_j = (x_j, y_j) \notin S$ can be tested easily to see whether p_j lies inside or outside of the m-contour. Any point that lies inside the m-contour is dominated by some maximal point. A simple search of the dynamic contour search tree would find a maximal point p_i that dominates p_j if p_j lies inside the m-contour. Otherwise, p_j will be found to lie outside the m-contour. Initialize v to the root. Let $p_i = (x_i, y_i)$ be the point at the leaf pointed to by *max_y(right(v))*. Figure 3 shows four quadrants with respect to p_i. The point p_j must lie in one of these quadrants. If $x_j > x_i$ and $y_j > y_i$ then the point p_j lies outside of the m-contour (quadrant I), as we shall show. If $x_j < x_i$ and $y_j < y_i$ then the point p_j is dominated by p_i and lies inside the m-contour (quadrant III). If $x_j < x_i$ and $y_j > y_i$ then recursively apply the search to v's left subtree (quadrant IV). If $x_j > x_i$ and $y_j < y_i$ then recursively apply the search to v's right subtree (quadrant II). If a leaf is reached, then p_j was not dominated by any point and thus lies outside the m-contour.

Theorem 2.2 Using the above procedure, an arbitrary point p_j will be found to lie inside or outside the m-contour in O($\log n$) time.

Proof: Suppose p_j lies outside the m-contour. No point dominates p_j so either p_j is found to lie clearly outside of the m-contour (quadrant I) or a leaf is reached during the search indicating that no points dominate p_j.

Suppose p_j lies inside the m-contour. Then there is at least one point on the m-contour which dominates p_j. Let p_k be the maximal point to the right of p_j that has the largest y-value. There is an internal node w in the dynamic contour search tree with its *max_y(right(w))* equal to p_k such that the leaf position in the tree that would be reached in searching for p_j is in w's left subtree. Thus, there is an internal node along the search path from the root to p_j whose *max_y(right(w))* value would determine that p_j is inside the m-contour. Figure 4a shows the point p_j dominated by three maximal points and Figure 4b shows the node w in the dynamic contour search tree.

Starting at the root of the tree, the search proceeds toward the node w. Let v be the current internal node in our search and let $p_i = (x_i, y_i)$ be the point at the leaf pointed to by *max_y(right(v))*. If p_i dominates p_j then our search stops. Otherwise, if p_i is a contour point, then w is either in v's left subtree if $x_k < x_i$ or in v's right subtree if $x_k > x_i$ and the search continues accordingly towards w. If p_i is not a contour point, then there are no contour points in v's right subtree. The node w is not in v's right subtree since if it was then p_i would be a contour point. Thus w is in v's left subtree and p_j's position in the tree is to the left of p_i. The search continues via the leftchild of v. It cannot happen that $p_i < p_j$, so the point p_j cannot lie in Quadrant I with respect to p_i. Thus either the search halts before reaching w, or it halts at w.

The search clearly takes $O(\log n)$ time. □.

3. INSERTIONS

Given the dynamic contour search tree for a set of n points, suppose we want to add an additional point p_j to the set. The point p_j will be inserted into the dynamic contour search tree. If p_j is outside the current m-contour, then p_j will become a new point on the m-contour, otherwise p_j will become an interior point. To maintain balance in the dynamic contour search tree after each insertion, we use the balancing scheme of the red-black binary search trees of Tarjan [3]. Each rotation performed during balancing will take $O(\log n)$ time to update the fields of the node involved. Thus it is crucial to use the red-black trees since they have $O(1)$ rotations per insertion.

To insert the point $p_j = (x_j, y_j)$ into the dynamic contour search tree, we use the insertion procedure for red-black trees, and update other fields as the insertion is performed. First search for the leaf position of p_j, a leaf node containing the point $p_i = (x_i, y_i)$. On the way down to find the insertion position, update *max_x(v)* and *max_y(v)* for each node v on the path by comparing it to p_j. Also on the way down, identify each node v such that *max_y(right(v))* will change to point to p_j. Each of these will have its *next_in_contour* pointer set to $NULL$ except for the one closest to the root.

For this node v, we compute its *next_in_contour* pointer as follows. The *next_in_contour* value of v should point to a leaf in v's left subtree. It can't point to a leaf in v's right subtree because then it would be pointing to a y-value which is larger than $y(max\_y(right(v)))$. To calculate v's *next_in_contour* pointer, traverse down a path in v's left subtree. At each internal node w, if $y(max\_y(right(w)))$ is greater than $y(max\_y(right(v)))$, then traverse down w's right subtree, else traverse down w's left subtree. If a leaf containing the point p_k is reached such that $y_k > y(max\_y(right(v)))$, then set v's *next_in_contour* pointer to point to this leaf. Otherwise set v's *next_in_contour* pointer to $NULL$ because there is no point in the subtree that is above and to the left in the plane.

As mentioned above, there may be several nodes v such that $max\_y(right(v))$ will change to point to p_j. If $max\_y(right(v))$ was previously pointing to p_k, then $transfer(w)$ for the leaf node w containing the point p_k is pointing to v as v must be the node closest to the root with a $max\_y(right(v)) = p_k$. $Next\_down(v)$ points to the next node u closest to the root with a $max\_y(right(u)) = p_k$, so when $max\_y(right(v))$ changes to point to p_j, $next\_down(v)$ is used to update $transfer(w)$ quickly. $Next\_down(v)$ will be updated as soon as the node it will point to is reached.

For nodes on the path between the root and the new leaf for p_j, their *next_in_contour* pointer could possibly change to p_j if p_j is in Quadrant IV with respect to them and p_j is closer than the point they are currently pointing to. A simple comparison at each node on the way down the path should tell whether the *next_in_contour* pointer should be changed to point to p_j.

Upon reaching p_i, replace p_i by an internal node and two leaves, one for p_j and one for p_i. The *transfer* pointer of the leaf p_j should be set to point to the internal node v which is the closest internal node to the root with a $max\_y(right(v))$ value of p_j.

Once p_j is inserted in a leaf, the insertion path is retraced updating the balance information. This will halt at some node with O(1) rotations performed. These rotations may affect three internal nodes that would need to recalculate their fields at a cost of $O(\log n)$.

In Figure 5a the change to the m-contour is shown when point p_9 is inserted into the set of points from Figure 1. The corresponding dynamic contour search tree is shown in Figure 5b.

Theorem 3.1 *Inserting a point p into a search tree with n leaves takes* $O(\log n)$ *time.*

Proof: To add the new point p_j involves a search from the root to a leaf which takes $O(\log n)$ time. At each internal node on the way down the path, updating *max_x* and *max_y* takes O(1) time. There is one *next_in_contour* pointer which must be calculated at a cost of $O(\log n)$ time. The remaining *next_in_contour* pointers that change are set to $NULL$ or are set to point to the leaf p_j and thus can be calculated in O(1) time. *Transfer* and *next_down* pointers are updated in O(1) time by adding or deleting a node from a linked list.

Balance information at each internal node can be calculated in O(1) time. Each rotation affects at most three internal nodes. Their fields can be recalculated in $O(\log n)$ time. Since there are O(1) rotations per insertion, the total time for balancing is $O(\log n)$ time. Thus the time to insert the point p_j into the search tree takes $O(\log n)$ time. □.

Corollary 3.1 *The search tree for a set of n points can be built in $O(n \log n)$ time.*

Proof: Apply the insertion procedure above for each point. □.

4. DELETIONS

Given the dynamic contour search tree for a set of n points, suppose we want to delete a point p_j from the set. If p_j was a point on the m-contour then the new m-contour will be updated when p_j is deleted by updating the dynamic contour search tree. Even if p_j was not on the m-contour, some work may be needed to update the dynamic contour search tree.

To delete the point $p_j = (x_j, y_j)$ from the dynamic contour search tree, we use the deletion procedure for the red-black trees. First find the leaf w containing the point p_j. Leaf w will be the child of some internal node v. Let u be the other child of v and let $p(v)$ be the parent of v. Remove w and v and let u become a child of $p(v)$. Now proceed back up the path, updating values at the internal nodes. Updating *max_x(v)* and *max_y(v)* at each internal node v along the path takes O(1) time. Updating *transfer* and *next_down* pointers also takes O(1) time per node along the path.

Some *next_in_contour* pointers may need to be recalculated. If any internal node v had *y(max_y(right(v)))* $= y_j$ then this node would receive a new *max_y(right(v))* value. Nodes with *y(max_y(right(v)))* $= y_j$ must lie along the search path from the root to p_j. If the *y(max_y(right(v)))* value of a node v changes, then the *next_in_contour* pointer of the node could possibly change, so its *next_in_contour* pointer must be recalculated. In addition, if any *next_in_contour* pointer was pointing at p_j, then these *next_in_contour* pointers need to be recalculated because p_j will be deleted. Only nodes along the path from the root to p_j can have *next_in_contour* pointers pointing to p_j since *next_in_contour* pointers must point to a descendant leaf. *Next_in_contour* pointers should be recalculated on the way back up to the root.

Also on the way back up the path, update the balance information using the balancing scheme of the red-black binary search trees. There may be O(1) rotations needed to balance the tree. At most three internal nodes will be affected by a rotation and need their fields recalculated. After performing a rotation, continue on up the path updating the remaining nodes on the path.

Figures 6a and 6b give a set of points and an associated dynamic contour search tree. Figures 7a and 7b give the set of points and the associated dynamic contour search tree after point p_9 is deleted from Figure 6.

Theorem 4.1 *Deleting a point from a search tree with n leaves takes* $O((\log n)^2)$ *time.*

Proof: To delete the leaf p_j from the dynamic contour search tree involves a search from the root to the leaf p_j which takes $O(\log n)$ time. After removing the point p_j, the same path must be traversed back up to the root, updating the fields at each internal node along the way. At most $O(\log n)$ nodes will need to recalculate *next_in_contour* pointers. It takes $O(\log n)$ time to recalculate one *next_in_contour* pointer so the total time in recalculating the *next_in_contour* pointers is $O((\log n)^2)$ time. Updating *max_x*, *max_y*, *transfer*, *next_down* and the balance information takes $O(1)$ time at each node. There will be $O(1)$ rotations and it will take $O(\log n)$ time to recalculate the fields of the nodes affected by the rotation. Thus the total time to delete point p_j from the search structure is $O((\log n)^2)$ time. □.

REFERENCES

1. KUNG, H. T., LUCCIO, F., AND PREPARATA, F. P. On finding the Maxima of a set of vectors. *J. ACM 22,* 4 (Oct. 1975), 469–476.
2. OVERMARS, M. H., AND VAN LEEUWEN, J. L. Maintenance of configurations in the plane. *Journal of Computer and System Sciences 23,* (1981), 166–204.
3. TARJAN, R. E. Updating a balanced search tree in O(1) rotations. *Inf. Proc. Lett. 16,* (1983), 253-257.
4. WILLARD, D. E., AND LUEKER, G. S. Adding range restriction capability to dynamic data structures. *J. ACM 32,* 3 (Jul. 1985), 597–617.

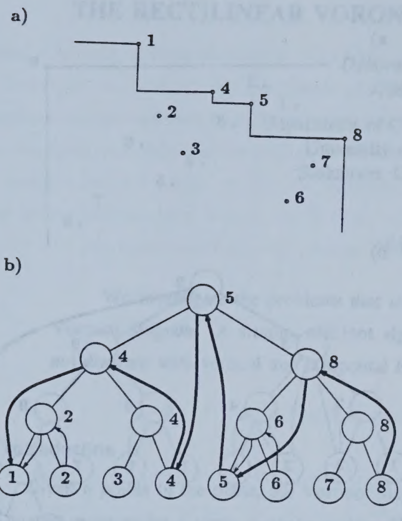

a)
1
4
5
2
8
3
7
6
b)
5
4
8
2
4
6
8
1
2
3
4
5
6
7
8

FIGURE 1

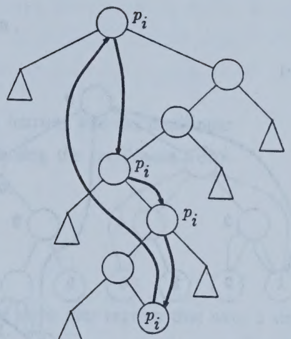


FIGURE 2

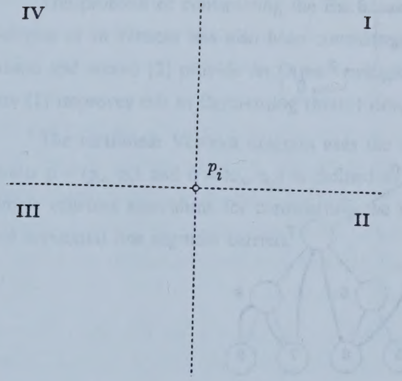

IV
I
p_i
III
II

FIGURE 3

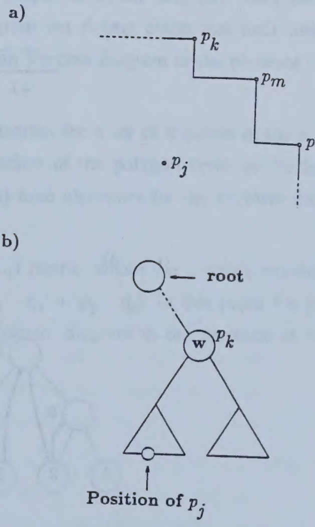

a)
p_k
p_m
p_n
p_j
b)
root
w
p_k
Position of p_j

FIGURE 4

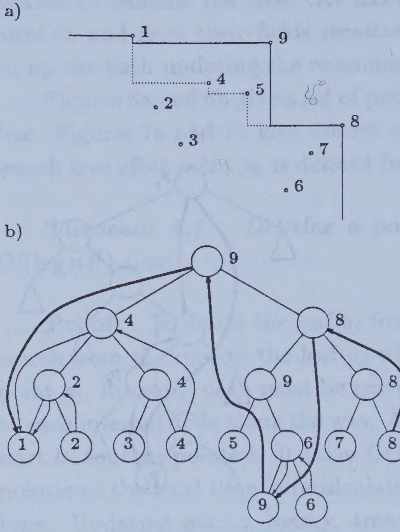

a)
1
9
4
5
2
3
8
7
6
b)
9
4
8
2
4
9
8
1
2
3
4
5
6
7
8
9
6

FIGURE 5

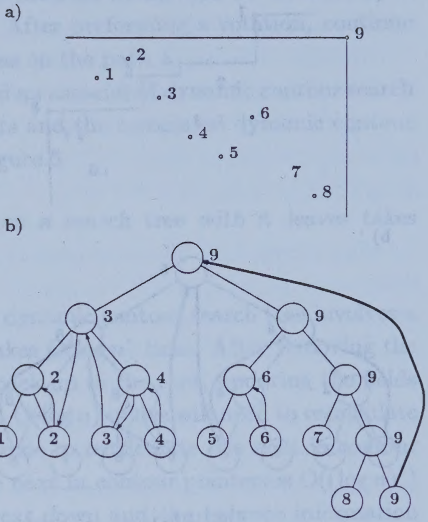

a)
9
2
1
3
6
4
5
7
8
b)
9
3
9
2
4
6
9
1
2
3
4
5
6
7
9
8
9

FIGURE 6

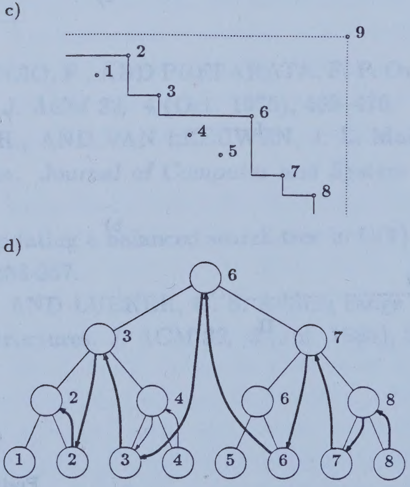

c)
9
2
1
3
6
4
5
7
8
d)
6
3
7
2
4
6
8
1
2
3
4
5
6
7
8

FIGURE 7

THE RECTILINEAR VORONOI DIAGRAM WITH BARRIERS

Deborah R. Fowler
J. Mark Keil

Department of Computational Science
University of Saskatchewan
Saskatoon, Canada S7N 0W0

ABSTRACT

We investigate the problems that arise by introducing barriers into the rectilinear Voronoi diagram. A simple, efficient algorithm for constructing the rectilinear Voronoi diagram with vertical and horizontal barriers is presented.

1. Introduction

Given n points in the plane, the Voronoi diagram partitions the plane into regions that have a single nearest point under a given distance metric. The Voronoi diagram has proven to be a useful tool in many contexts in computational geometry and one may wish to extend the applicability of the Voronoi diagram to a domain which contains barriers to travel. These barriers occur naturally in application areas such as robotics and VLSI design.

The idea of constructing the Euclidean Voronoi diagram in the presence of line segment barriers was first considered by Franklin et al [6]. They note that, with the Euclidean distance metric, edges of the diagram may be hyperbolic sections and discuss other properties of the diagram. They provide a general inductive method for constructing the Voronoi diagram but do not claim that their method is efficient. The problem of efficiently constructing the Euclidean Voronoi diagram in the presence of arbitrary line segment barriers remains open.

The problem of constructing the Euclidean Voronoi diagram for a set of n points inside a simple polygon of m vertices has also been considered. Here the edges of the polygon serve as the barriers. Asano and Asano [2] provide an O(mn + mloglogm + nlogn) time algorithm for the problem and Aronov [1] improves this to $O((m+n)\log^2(m+n))$ time.

The rectilinear Voronoi diagram uses the rectilinear (L_1) metric, where the distance between two points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ is defined as $d(p,q) = |p_x - q_x| + |p_y - q_y|$. In this paper we provide simple efficient algorithms for constructing the rectilinear Voronoi diagram in the presence of vertical and horizontal line segment barriers.

2. Vertical Line Segment Barriers

We first consider the problem of constructing the rectilinear Voronoi diagram of a set of n points in the presence of m vertical line segment barriers. In the L_1 metric, the bisector between two points in the presence of m vertical barriers can be a polygonal line made up of O(m) line segments as in Figure 1. To avoid the explicit construction and intersection of these bisectors we will partition the plane into regions which do not contain barriers and separately compute the portion of the Voronoi diagram in each of these regions. In particular, we will partition the plane by the vertical lines that contain the barriers. These dividing lines will be called slab lines and the regions between these lines slabs.

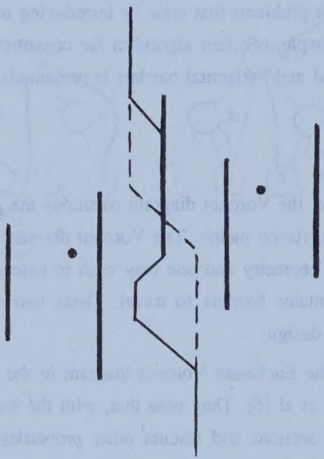


Figure 1 Bisector with Vertical Barriers

The portion of the Voronoi diagram with barriers that lies within a particular slab may contain regions corresponding to points outside the slab. For example in Figure 2, point p_1 influences the portion of the Voronoi diagram that lies in the leftmost slab. Thus to form the correct portion of the Voronoi diagram that lies in a slab points outside the slab are represented by new representative points on the vertical slab lines. These representative points are given weights according to the distance they are from the point they represent and the original points lying in the slab are given weights of zero. This weighting of the points enables us to reduce the problem of computing the portion of the Voronoi diagram in the slab to that of computing an additive weight Voronoi diagram on the points in the slab and the representative points on the slab boundary. In the additive weight Voronoi diagram the metric is the sum of the weight of a point and the rectilinear distance to the point.

Fortune [3] recently developed a sweepline algorithm for constructing the Euclidean Voronoi diagram in O(nlogn) time. He also adapted the technique to handle points with additive weights [3] and convex distance functions [4]. Fowler [5] filled in the detail necessary to use this algorithm, with the rectilinear metric and points with additive weights, in O(nlogn) time.

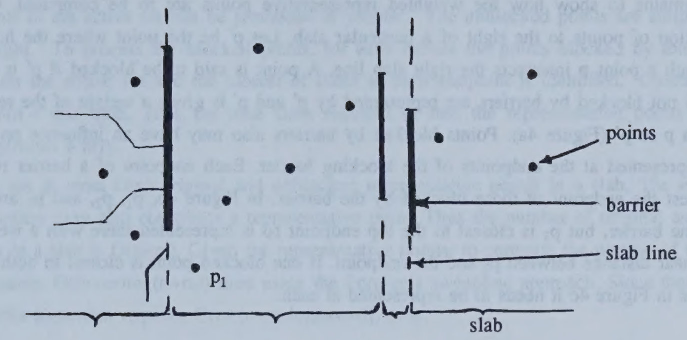


Figure 2 Slab lines and Slabs

Thus to construct the Voronoi diagram with vertical barriers, we first partition the plane into slabs. Then for each slab we find weighted representative points for points outside the slab that will influence the diagram in the slab. Finally, we use the sweepline algorithm on the original and weighted representative points in the slab to form the Voronoi diagram. Figure 3 shows a rectilinear Voronoi diagram with vertical barriers.

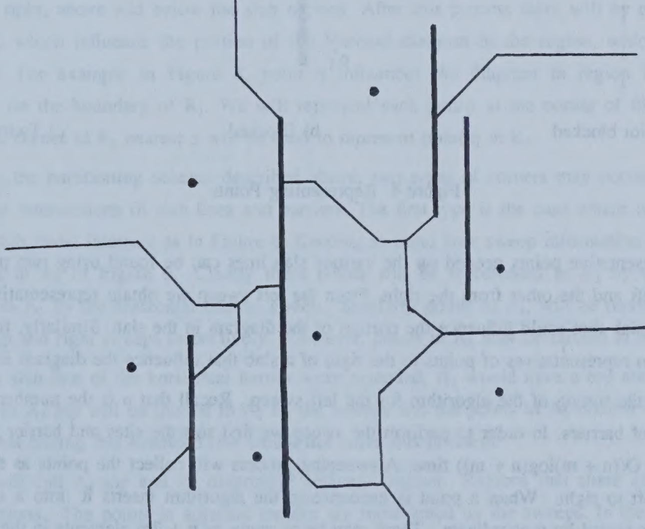
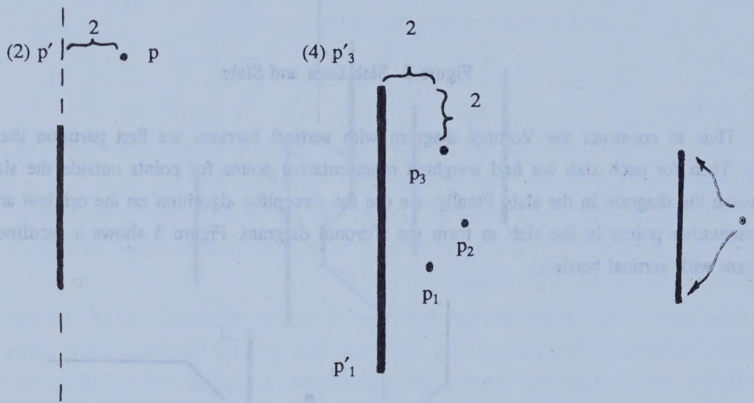


Figure 3 Rectilinear Voronoi Diagram with Vertical Barriers

It remains to show how the weighted representative points are to be computed. Consider the representation of points to the right of a particular slab. Let p′ be the point where the horizontal line through such a point p intersects the right slab line. A point is said to be blocked if p′ is on a barrier. The points not blocked by barriers are represented by p′ and p′ is given a weight of the rectilinear distance from p to p′ (Figure 4a). Points blocked by barriers also may have an influence on the diagram and are represented at the endpoints of the blocking barrier. Each endpoint of a barrier represents the point nearest the endpoint of those blocked by the barrier. In Figure 4b, p_1, p_2, and p_3 are all blocked by the same barrier, but p_3 is closest to the top endpoint so is represented there with a weight equal to the rectilinear distance between p_3 and the endpoint. If one blocked point is closest to both endpoints of a barrier as in Figure 4c it needs to be represented at each.



a) Not blocked b) Blocked c) Twice

Figure 4 Representing Points

The representative points needed on the various slab lines can be found using two plane sweeps, one from the left and the other from the right. From the left sweep we obtain representatives of points to the left of a slab that could influence the portion of the diagram in the slab. Similarly, from the right sweep we obtain representatives of points to the right of a slab that influence the diagram in the slab.

Consider the timing of the algorithm for the left sweep. Recall that n is the number of sites and m the number of barriers. In order to perform the sweep we first sort the sites and barrier endpoints by x-coordinate in $O((n + m)\log(n + m))$ time. A sweeping process will collect the points as the sweepline sweeps from left to right. When a point is encountered the algorithm inserts it into a set of "active points" which is sorted by y-coordinate. There may be as many as $n + 2m$ elements in the set of active points as representative points have barrier endpoints which may need to be included. Altogether, inserting points into the active set may require $O((n + m)\log(n + m))$ time.

At a slab line, the weighted representative points for the slab must be calculated, inserted into the representative list for that slab line, and then continued to be represented in non-adjacent slabs. To do this the points in the active set can be processed in y-order. The unblocked points are simply given an updated weight. To process the blocked points, for each barrier the points blocked by the barrier are removed from the active set and the closest of these to each endpoint is identified. Overall this may require O(m(n + m)) time. Thus the total time required to find the representative points is at most O((n+m)[log(n+m) + m]).

There are at most O(n) original and unblocked representative points in a slab. The endpoints of the O(m) barriers may also contribute a representative point. Thus the number of original and representative points in a slab is O(n+m). Given the representative points, to compute the portion of the diagram in a slab requires O((n+m)log(n+m)) time using the Fortune's sweepline approach. Since there are O(m) slabs the entire algorithm requires O(m[(n+m)log(n+m)]) time.

3. Vertical and Horizontal Barriers

In this section we will extend our rectilinear Voronoi diagram algorithm to handle both horizontal and vertical disjoint line segment barriers. We use the same approach of partitioning the plane into slab regions which do not contain barriers. The plane is first partitioned by introducing the vertical slab lines that contain vertical barriers. It is further partitioned by introducing the smallest horizontal line segments which contain the horizontal barriers and whose endpoints lie on the vertical slab lines.

To find the representative points for a slab region we generalize the approach used with the vertical barriers and perform four plane sweeps, left, right, up and down to gather information about points to the left, right, above and below the slab regions. After this process there will be points outside the slab region, which influence the portion of the Voronoi diagram in the region, which have not been represented. For example in Figure 5, point q influences the diagram in region R_1 but it is not represented on the boundary of R_1. We will represent such points at the corner of the region, thus in Figure 5 the corner of R_1 nearest q will be used to represent point q in R_1.

Using the partitioning scheme described above, two types of corners may occur: intersections of slab lines or intersections of slab lines and barriers. The first type is the case where two extended barrier lines (slab lines) intersect as in Figure 6. Keeping in mind how sweep information is obtained, consider points in A_2 of Figure 6. Clearly these points will be represented in A_3 by the vertical (left) sweep and in A_1 by the horizontal (down) sweep. Similarly points in A_1, will be represented in A_2 and A_3 by the up and right sweeps respectively. However, points in A_3 may be missed in A_1 and A_2.

If the slab line of the horizontal barrier were extended, A_3 would have a top and bottom portion. The points in A_3 top will be missed in A_1 by the sweeps and the points in A_3 bottom will be missed in A_2. Note that having this extended line would **not** solve this problem.

We will call A_3 top and A_1 diagonally adjacent regions. Regions that share an edge are called adjacent regions. The points in adjacent regions are represented by the sweeps. In the diagonally adjacent regions points which may affect the region being considered may not be represented by the regular sweeps, and must be represented at the corner point. A similar situation may occur with the second type of intersection (Figure 7).

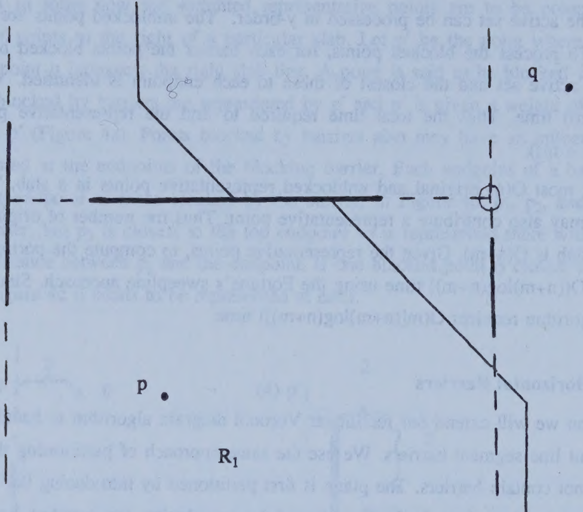


Figure 5 Corner Representation

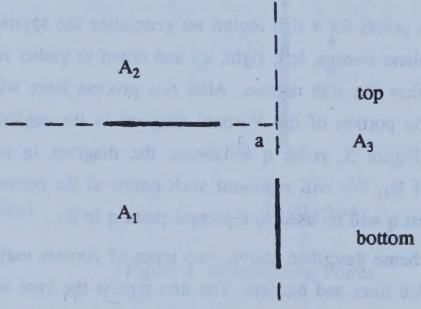


Figure 6 Slab line Intersections

Representing Points in Diagonally Adjacent Regions by Corner Points

Any rectangular slab region will have at most four corners (regions of the outside of the partition may have only one or two), of either type of intersection.

First look at slab line intersections. Consider the case of the top, right corner of R_1 in Figure 8. In R_1 there may be missed points in the diagonally adjacent region "this area". Any point from the diagonally adjacent region will have to travel through corner point "a" to get to R_1 by the shortest route. Only one of the "missed points" needs to be represented at the corner point "a"; the one whose total distance to the corner is the smallest. This can be found by examining the right list (the list of representative points at the given slab line created during the right sweep) and taking the closest one from the diagonally adjacent region to the corner. By closest, we mean the weight of the point on the list (from the y-level of "a" (a_y) upward) plus the distance to the corner.

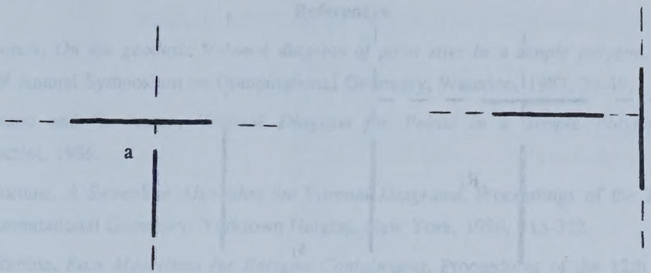


Figure 7 Barrier/Slab line Intersections

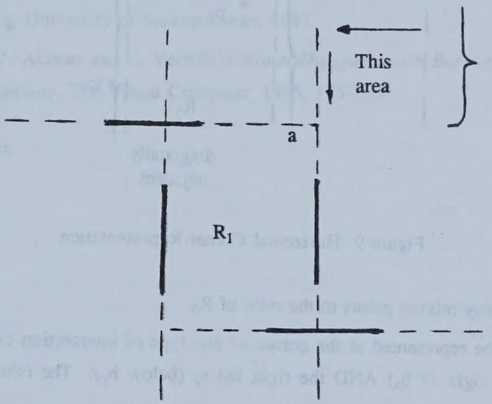


Figure 8 Corner Representation

Now what about slab line/ barrier corners? Consider R_1 and corner "b" in Figure 9. The points to the right of R_1 and above b_y will be represented on the right slab segment of R_1. The points below R_1 and to the left of b_x will be represented in the bottom slab segment list. Note that the point to be represented at the corner is not just the point represented at the endpoint of the barrier in the bottom sweep; for example p_1 or p_2 in Figure 9 could be closer. We must examine what points might be missed. Points from the diagonally adjacent region (R_2) may not be represented in the sweep, so the slab line s_3 must be examined. Also, the points directly to the right of the diagonally adjacent region may be missed because the horizontal line stops at the vertical slab line. They are represented at the vertical slab lines, s_2 and s_1. We do not want to examine the right list s_1 (directly below b_y) for two reasons. One, because a point represented on s_1 coming from R_2 many not have to go around the barrier (for example point p_1 in Figure 9). Secondly, the points in R_2 are represented at s_3 (obtained from the upward sweep). Since the points in R_2 are being represented on s_3, we need only examine s_2

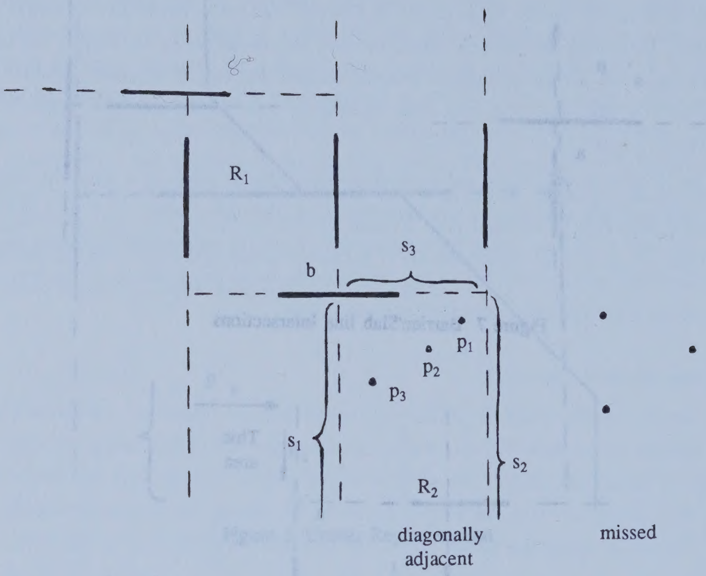


Figure 9 Horizontal Corner Representation

(below b_y) to identify any missed points to the right of R_2.

So the point to be represented at the corner of this type of intersection can be found by examining the list s_3 (to the right of b_x) AND the right list s_2 (below b_y). The other corners follow analogously.

It should be noted that it is possible to incorporate the computation of the representative points at the corners of regions into the four plane sweeps without increasing the time bound. Each slab region may contain O(n+m) original or representative points and there may be $O(m^2)$ slab regions so that in the worst case our approach yields an $O(m^2[(m+n)\log(m+n)])$ time algorithm for computing the Voronoi diagram of a set of n points in the presence of m disjoint vertical and horizontal line segment barriers.

References

[1] B. Aronov, *On the geodesic Voronoi diagram of point sites in a simple polygon*, Proceedings of the 3rd Annual Symposium on Computational Geometry, Waterloo, 1987, 39-49.

[2] T. Asano and T. Asano, *Voronoi Diagram for Points in a Simple Polygon*, unpublished manuscript, 1986.

[3] S.J. Fortune, *A Sweepline Algorithm for Voronoi Diagrams*, Proceedings of the 2nd Symposium on Computational Geometry, Yorktown Heights, New York, 1986, 313-322.

[4] S.J. Fortune, *Fast Algorithms for Polygon Containment*, Proceedings of the 12th Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 194, Springer-Verlag, New York, 189-198.

[5] D.R. Fowler, *The Rectilinear Voronoi Diagram with Barriers*, M.Sc. Thesis, Department of Computational Science, University of Saskatchewan, 1987.

[6] W.R. Franklin, V. Akman and C. Verrilli, *Voronoi Diagrams with Barriers and on Polyhedra for Minimal Path Planning*, The Visual Computer, 1985, 133-150.

SOME RESTRICTED RECTANGLE COVERING PROBLEMS

HAROLD E. CONN and JOSEPH O'ROURKE
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

ABSTRACT

We consider some restricted versions of the problem of finding a minimum cover of an orthogonal polygon by a set of rectangles, an NP-complete problem. The problem remains NP-complete for orthogonal polygons in general position, and for restricted versions where only the perimeter segments of the polygon must be covered by the rectangles, and where only the reflex vertices of the polygon must be covered by the rectangles. Polynomial algorithms are given for finding a minimum cover of the convex vertices, and for finding a minimum cover of the horizontal perimeter segments of a general position orthogonal polygon.

1. INTRODUCTION

The problem of finding a minimum rectangle cover of an orthogonal polygon has received considerable attention in the literature [1], [2], [3]. An *orthogonal polygon* (OP) is a polygon having all its edges parallel to either of two given orthogonal directions. It may contain holes that are themselves OPs. A *rectangle cover* of an OP is a collection of rectangles each of which is a subset of the OP and such that the set union of the collection of rectangles is equal to the OP. A cover is a *minimum* cover (MC) if there is no other cover containing fewer rectangles.

In this paper we analyze the complexity of the minimum rectangle cover problem under certain restrictions that have not been previously considered. We know of no application for our results; our motivation is simply to better understand covering problems by further exploring the boundary between those that are known to be NP-complete and those for which polynomial algorithms are known. Some of these restrictions represent weakenings of the minimum rectangle cover problem, but we show that the new problems remain NP-complete. Further weakenings result in several polynomial-time solvable covering problems. In addition, several open problems are posed.

A *maximal rectangle* is a rectangle that is a subset of a given OP and is not a subset of any other rectangle contained in the polygon. In trying to find a minimum rectangle cover of an OP, no advantage is gained by considering any rectangle that is not maximal, so henceforth the term *rectangle* should be taken to mean maximal rectangle, and the term *cover* should be taken to mean a cover by maximal rectangles.

If the OP is not subject to any convexity restrictions and is allowed to contain holes, Masek proved that the problem of finding a MC is NP-complete [1]. The case of a convex OP was analyzed by Chaiken *et al.* An OP is *convex* if any line parallel to either of the two reference orthogonal directions intersects the polygon in at most one line segment. They proved that the size of a minimum rectangle cover, θ, is equal to the size of a maximum anti-rectangle, α. An *anti-rectangle* is a set of points in the polygon no two of which are contained in the same rectangle. Their proof that $\theta = \alpha$ serves as the basis for a polynomial-time algorithm for finding a MC of a convex OP [2]. In general it appears that if $\theta = \alpha$, the problem is easier, and a polynomial-time algorithm is possible. Later Győri proved that $\theta = \alpha$ for any OP that is monotone (convex in one of the two given orthogonal directions) [4]. Franzblau and Kleitman also proved this and gave a $O(n^2)$ algorithm for finding a MC of a monotone OP [3]. The complexity of the problem of finding a MC of a simply-connected (no holes) OP with no other convexity restrictions remains unknown.

The polygons considered in this paper contain holes and are not subject to any convexity restrictions. The first three restrictions introduced involve what must be covered. In the first restriction, instead of requiring that the interior of the OP be covered, we say that a collection of rectangles is a *perimeter* cover if all of the perimeter segments of the OP (including the boundaries of any holes) are covered by the collection of rectangles. In general, fewer rectangles are required to cover the perimeter than the whole polygon. The polygon in Figure 1 requires five rectangles to cover the whole polygon, but only four (shaded) are needed to cover the perimeter. We also consider covering only those perimeter segments that are parallel to a given direction (we arbitrarily choose horizontal).

The second restriction is to require that the collection of rectangles must cover only the notches of the polygon. The *notches* of an OP consist of the reflex vertices (those with an internal angle of 270 °) of the polygon and the convex vertices of any holes in the polygon. The definition of covering a notch is given in Section 3.

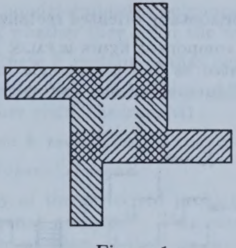


Figure 1

The third restriction is to require that the collection of rectangles must cover only the corners of the polygon. The *corners* of an OP consist of the convex vertices of the polygon and the reflex vertices of any holes in the polygon. A corner is *covered* by a rectangle if that corner coincides with one of the corners of the rectangle.

The fourth restriction considered involves collinearities among the perimeter segments of the OP. The perimeter of an OP consists of four different *types* of perimeter segments: top, bottom, left, and right. If the polygon contains holes, the perimeter segments surrounding each hole are classified exactly opposite to the way they would be if the hole were a free-standing OP. We say that an OP is in *general position* if no two perimeter segments of the same type are internally collinear: that is, there is no line segment contained entirely within the polygon that is collinear with the two perimeter segments, and intersects both segments. Thus when an OP is in general position, no rectangle can cover more than one perimeter segment of a given type. The class of general position polygons is a subclass of all polygons, and it is possible that problems posed for this restricted class are easier.

Our results are summarized in Table 1. The complexity of finding a minimum rectangle cover of the horizontal perimeter segments of an OP not in general position remains an open problem.

Table 1
Complexity of Covering Problems

Cover Type	General Position	Not General Position
Interior	NP-complete	NP-complete [1]
Perimeter Segments	NP-complete	NP-complete
Notches	NP-complete	NP-complete
Horizontal Perimeter Segments	$O(n^{2.5})$	?
Corners	$O(n^{2.5})$	$O(n^{2.5})$

2. COVERING THE PERIMETER

Masek proved that finding a MC of an OP not in general position is an NP-complete problem [1]. His proof uses a transformation from the problem of satisfiability of Boolean expressions. It uses an OP to simulate the behavior of a Boolean expression. The OP is built up from component figures that simulate various Boolean connectives.

In this section it is shown that the problem remains NP-complete even when restricted to finding a minimum cover of the *perimeter* of an OP, where the perimeter is defined to include the boundary of any holes in the polygon. Our proof of NP-completeness for the perimeter cover problem operates in the same way as Masek's except that some of the component figures have to be modified. Before analyzing the problem of covering the perimeter, Masek's results are summarized.

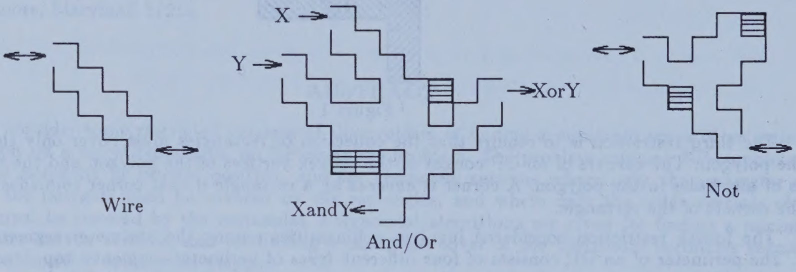
2.1 Covering the Interior of an Orthogonal Polygon

Given an OP which may contain orthogonal holes, we wish to find an optimal cover of the polygon by maximal rectangles. Masek proved that this problem is NP-complete by a transformation from the problem of satisfiability of Boolean expressions, LO7 in [5]. The Boolean expressions are composed using three connectives: NOT , AND , and OR.

An arbitrary Boolean expression is simulated by an OP. The polygon is constructed from component figures that can be covered efficiently iff they perform like the corresponding Boolean expression connectives. The component figures are laid out on a grid of unit squares. Figure 2 displays the component figures used in the simulation.

The truth values of Boolean variables in the expression are propagated through 'wire' figures. In wires and at connection points between figures, TRUE values are represented by 2x1 vertically oriented rectangles, and FALSE values by 1x2 horizontally oriented rectangles.

If an input to or output from a component figure is FALSE , then only half a rectangle is covered at the connection point. This is counted as half a rectangle in an optimal cover of the component figure. To efficiently cover an entire figure, all of the half-rectangles in the figure must be combined with other half-rectangles.



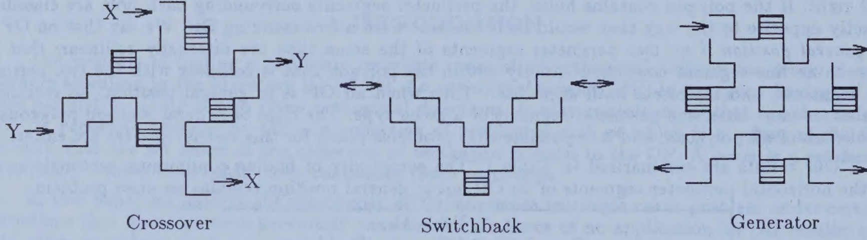


Figure 2

The following lemmas link the behavior of the component figures (in simulating the components of Boolean expressions) with the minimum covers of the component figures.

Lemma M1: A length k 'wire' figure can be covered with k rectangles iff its input and output values are identical. (Length of wire = number of bends in figure.)

Lemma M2: A 'not' figure can be covered with 5 1/2 rectangles iff the values on the wires connected to it are complementary.

Lemma M3: An 'and/or' figure can be covered with 12 rectangles iff the 'or' output value is the logical OR of the inputs and the 'and' output value is the logical AND of the inputs.

Lemma M4: A 'generator' figure can be covered with 11 rectangles iff all of the wires connected to the generator have the same value.

Lemma M5: A 'crossover' figure can be covered with 9 rectangles iff the truth values of the opposing wires are identical.

Lemma M6: A 'switchback' figure can be covered with 7 rectangles iff the truth values at both ends are identical.

A TRUE component figure can be represented by a 'wire' figure of length 4 as shown in Figure 2. A FALSE component figure is like the 'wire' in Figure 2 except with the last unit square at the bottom of the wire erased.

Lemma M7: A 'true' ('false') figure can be covered with 4 (3 1/2) rectangles iff the input to the figure is TRUE (FALSE).

An algorithm is given in [1] that constructs, in polynomial time, an OP that simulates an arbitrary Boolean expression and its dual. After constructing such an OP F, one can calculate by means of Lemmas M1-M7 exactly how many rectangles $E(F)$ are required for an efficient cover of F. It may or may not be possible to cover F with that number. Let $M(F)$ denote the number of rectangles required to cover F.

Lemma M8: Let F be the OP constructed to simulate an arbitrary Boolean expression. Then that Boolean expression is satisfiable iff $E(F)=M(F)$.

The problem can be solved in nondeterministic polynomial time by first guessing a set of k maximal rectangles and then checking whether they cover the whole OP.

Proposition M1: The problem "are there k rectangles that cover an arbitrary OP?" is in NP.

Since it is known that the problem of satisfiability of Boolean expressions is NP-complete, Lemma M8 and Proposition M1 together yield Theorem M1.

Theorem M1: The problem "are there k rectangles that cover an arbitrary OP?" is NP-complete.

2.2 Covering the Perimeter of an Orthogonal Polygon

Now we consider the complexity of the restricted problem "are there k rectangles that cover the *perimeter* of an arbitrary orthogonal polygon?". This problem can also be shown to be NP-complete by a transformation from the problem of Boolean satisfiability. The transformation mimics that used in [1].

Masek defines the concept of the *atomic subareas* of an OP [1]. These are the areas obtained by extending a horizontal ray and a vertical ray from each notch of the OP to the first edge of the polygon hit by the ray. Note that some atomic subareas may be completely internal to the OP while others will share a finite length portion of its perimeter. Those that share some of the perimeter will be referred to as *perimeter-adjacent* atomic subareas. An OP is covered by a collection of maximal rectangles iff all its atomic subareas are covered by the collection. We define an atomic subarea to be *essential* if there exists only one maximal rectangle that covers it, and we say that a rectangle is *perimeter-essential* if it is the only rectangle that covers some perimeter-adjacent atomic subarea. Figure 2 displays the component figures with their essential atomic subareas shaded.

As in [1], an arbitrary Boolean expression is simulated by an orthogonal polygon. The polygon is built up from component figures having *perimeters* that can be covered efficiently iff the figures perform like the corresponding Boolean expression connectives. The component figures used are identical to those used in [1] except for the 'and/or' figure. The following Lemma is of use in determining whether the component figures perform as desired.

Lemma 1: If every atomic subarea of an OP is either perimeter-adjacent or is covered by a perimeter-essential rectangle, then any MC of the perimeter is also a MC of the whole polygon.

Proof: The proofs of this and the following lemmas are omitted. They are given in [6].

For an example of the use of Lemma 1, see [6]. Using Lemma 1 we can see from Figure 2 that the figures for 'wire', 'not', 'generator', 'crossover', 'switchback', 'true', and 'false' behave as desired, since the lemma guarantees that any MC of the perimeter of any of these figures is also a MC of the whole figure. Thus the perimeter of each can be covered efficiently iff the figure acts like the corresponding Boolean connective. Thus we can state analogs to Lemmas M1, M2, and M4-M7.

Lemma 2.1: The perimeter of a length k 'wire' figure can be covered with k rectangles iff its input and output values are identical. (Length of wire = number of bends in figure.)

Lemma 2.2: The perimeter of a 'not' figure can be covered with 5 1/2 rectangles iff the values on the wires connected to it are complementary.

Lemma 2.4: The perimeter of a 'generator' figure can be covered with 11 rectangles iff all of the wires connected to the generator have the same value.

Lemma 2.5: The perimeter of a 'crossover' figure can be covered with 9 rectangles iff the truth values of the opposing wires are identical.

Lemma 2.6: The perimeter of a 'switchback' figure can be covered with 7 rectangles iff the truth values at both ends are identical.

Lemma 2.7: The perimeter of a 'true' ('false') figure can be covered with 4 (3 1/2) rectangles iff the input to the figure is TRUE (FALSE).

If we attempt to apply Lemma 1 to the 'and/or' figure there is one atomic subarea, call it S, that does not satisfy either of the two criteria of the Lemma, as illustrated in Figure 3.

Thus the 'and/or' component figure might not behave as desired with respect to perimeter covers. In fact, it does not. It behaves as desired for three of the four possible input combinations. There are two inputs to the 'and/or' figure, one above the other. Let the higher input be designated X, and the lower Y. When the X input carries a value of FALSE and the Y input carries a value of TRUE, an ambiguity arises. There are two different minimum perimeter covers possible. One case results in the correct truth values being output, while the other inverts them resulting in an AND output value of TRUE and an OR output value of FALSE. In the latter case the atomic subarea S is not covered, but since it does not include any of the perimeter of the polygon, it need not be covered.

Since the 'and/or' figure cannot be used, and since the Boolean expressions to be simulated may contain AND and OR connectives, some other component figure(s) must be found to complete the simulation. The difficulty with the 'and/or' connective arises when the X input carries a FALSE value and

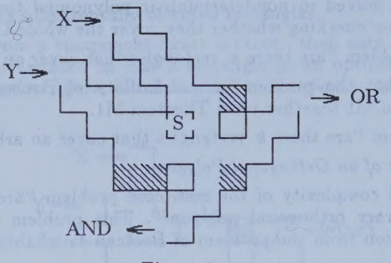


Figure 3

the Y input carries TRUE. There are two possibilities for the minimum perimeter cover in this situation. Although one of the cases results in outputs exactly opposite to those desired, the other case does yield the correct outputs. Thus it seems possible that we may be able to 'force' the correct output values to always occur in some component figure simulating a double connective.

The 'and/or' component figure serves as the basis for the 'double and'. Two copies of the 'and/or' are used: one oriented normally, the other being a mirror image about the y-axis. The OR outputs of the two copies are connected together by means of a 'switchback' figure. Since a MC of a 'switchback' figure requires truth values at both ends to be identical, if either side of the 'switchback' carries the correct OR output value it should resolve any ambiguity on the other side in favor of the correct value. We can be sure that one of the OR output values is always correct by inverting the order in which the two inputs are fed into the two copies of the 'and/or' figure. Construction of a 'double or' component figure is analogous. See Figure 4.

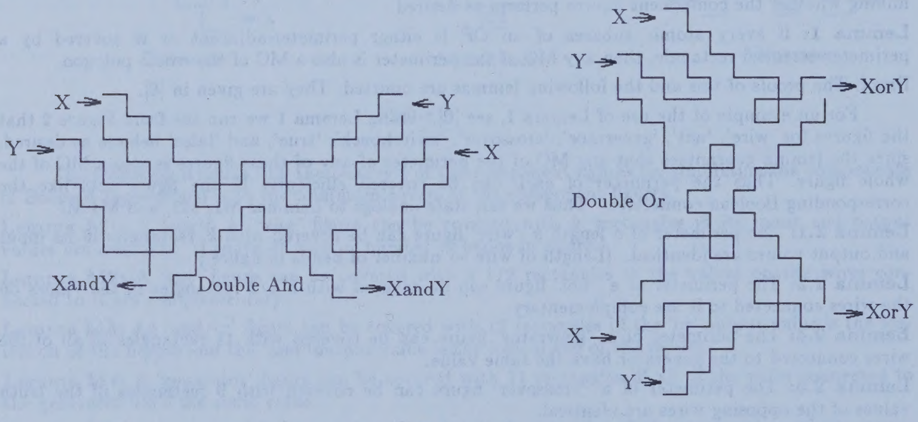


Figure 4

Lemma 2.3: The perimeter of a 'double and' figure can be covered with 23 rectangles iff the AND output values are the logical AND of the X and Y inputs. The perimeter of a 'double or' figure can be covered with 24 rectangles iff the OR output values are the logical OR of the X and Y inputs.

The set of logical connectives is now complete, and we are able to construct an OP to simulate an arbitrary Boolean expression. The algorithm for constructing the OP is essentially the same as that given in [1], except that every 'and/or' figure is replaced by either a 'double and' or a 'double or' as appropriate. The algorithm still runs in polynomial time. See [6] for an analysis.

After constructing the required OP F, one can calculate by means of Lemmas 2.1-2.7 exactly how many rectangles $EP(F)$ are required for an efficient cover of the perimeter of F. It may or may not be possible to cover the perimeter with that number of rectangles. Let $MP(F)$ denote the number of rectangles required to cover the perimeter of F.

Lemma 3: Let F be the OP constructed to simulate an arbitrary Boolean expression. Then that Boolean expression is satisfiable iff $EP(F)=MP(F)$.

Clearly the problem is in NP, and thus we have Theorem 1.

Theorem 1: The problem "are there k rectangles that cover the perimeter of an arbitrary OP?" is NP-complete.

3. COVERING NOTCHES

The *notches* of an OP consist of the reflex vertices of the polygon and the convex vertices of any holes in the polygon. Notches (and corners) are points at the intersection of two orthogonal perimeter segments of the polygon. We say that a notch is *covered* by rectangles if there exists an $\epsilon>0$ such that every point on each of the two perimeter segments defining the notch, within distance ϵ of the notch, is covered by a rectangle. Thus in order to cover any notch two rectangles are required. In general, fewer rectangles are needed to cover the notches of an orthogonal polygon than are needed to cover the perimeter. See [6] for an example.

In this section we show that finding a minimum rectangle cover of the notches of an OP is an NP-complete problem. The transformation is again from the problem of satisfiability of Boolean expressions, and the component figures used are identical to those used in the construction for the perimeter cover problem. We have only to show that the component figures retain the desired attributes of Boolean connectives with respect to finding minimum notch covers.

We say that an atomic subarea is *notch-adjacent* if it includes points on the perimeter of the polygon that are arbitrarily close to a notch, and a rectangle is *notch-essential* if it is the only rectangle that covers some notch-adjacent atomic subarea. The following lemma is of use in determining whether the component figures perform as desired with respect to notch covers.

Lemma 4: If every atomic subarea of an OP is either notch-adjacent or is covered by a notch-essential rectangle, then any MC of the notches of the polygon is also a MC of the whole polygon.

Using Lemma 4 we can see from Figure 2 that the figures for 'wire', 'generator', 'crossover', 'switchback', 'true', and 'false' behave as desired, since the lemma guarantees that any minimum cover of the notches of any of these figures is also a MC of the whole figure. Thus the notches of each of these figures can be covered efficiently iff the figure performs like the corresponding Boolean connective. Thus we can state analogs to Lemmas M1 and M4 - M7.

Lemma 5.1: The notches of a length k 'wire' figure can be covered with k rectangles iff its input and output values are identical. (Length of wire = number of bends in figure.)

Lemma 5.4: The notches of a 'generator' figure can be covered with 11 rectangles iff all of the wires connected to the generator have the same value.

Lemma 5.5: The notches of a 'crossover' figure can be covered with 9 rectangles iff the truth values of the opposing wires are identical.

Lemma 5.6: The notches of a 'switchback' figure can be covered with 7 rectangles iff the truth values at both ends are identical.

Lemma 5.7: The notches of a 'true' ('false') figure can be covered with 4 (3 1/2) rectangles iff the input to the figure is TRUE (FALSE).

We cannot apply Lemma 4 to the 'not' figure because there is one atomic subarea that is neither notch-adjacent nor covered by a notch-essential rectangle. Still, the 'not' figure does retain the desired relationship between minimum notch covers and its behavior in simulating the Boolean connective NOT. There are three possible minimum notch covers of the 'not' figure. Each requires 5 1/2 rectangles and is such that the truth values on the wires are complementary. See [6].

Lemma 5.2: The notches of a 'not' figure can be covered with 5 1/2 rectangles iff the truth values on the wires connected to it are complementary.

The 'and/or' figure also fails to meet the hypothesis of Lemma 4 since the atomic subarea labelled S in Figure 3 is neither notch-adjacent nor covered by a notch-essential rectangle. Furthermore the 'and/or' figure does not behave as desired with respect to notch covers for the same reasons that it failed with respect to perimeter covers as explained in Section 2. To complete the set of connectives, we will use the 'double and' and 'double or' figures of Section 2. The following lemma is of use in verifying that they retain the desired behavior with respect to notch covers.

Lemma 6: For any OP, if every atomic subarea that includes some part of the perimeter of the polygon is either notch-adjacent or is covered by a rectangle that is notch-essential, then any MC of the notches of the polygon is also a MC of the perimeter of the polygon.

By examining Figure 4 we can see that the 'double and' and 'double or' each contain only two atomic subareas that are not notch-adjacent and are not covered by a notch-essential rectangle. (These correspond to the atomic subarea labelled S in Figure 3.) Since these atomic subareas do not include any of the perimeter of the figures, we can apply Lemma 6 to the 'double and' and 'double or', and with the use of Lemma 2.3 regarding perimeter covers, we conclude Lemma 5.3.

Lemma 5.3: The notches of a 'double and' figure can be covered with 23 rectangles iff the AND output values are the logical AND of the X and Y inputs. The notches of a 'double or' figure can be covered with 24 rectangles iff the OR output values are the logical OR of the X and Y inputs.

Clearly the problem is in NP, so we have Theorem 2.

Theorem 2: The problem "are there k rectangles that cover the notches of an arbitrary OP?" is NP-complete.

In the next section we show that a MC of the corners of an OP can be found in polynomial time. Thus we have the following corollary to Theorem 2:

Corollary 1: The problem "are there k rectangles that cover all the vertices (convex and reflex) of an arbitrary OP?" is NP-complete.

4. COVERING CORNERS

The *corners* of an OP consist of the convex vertices of the polygon and the reflex vertices of any holes in the polygon. A corner is *covered* by a rectangle if it coincides with one of the corners of the rectangle. An example was given in [2] to show that $\theta_C \neq \alpha_C$ for non-convex OPs; that is, the size of a minimum corner cover, θ_C, may not be equal to the size of a maximum anti-rectangle, α_C, where the anti-rectangle points must be corners of the polygon. Still, a minimum corner cover can be found in polynomial time by reformulating the problem into one of finding a maximum matching in a graph.

The nodes of the graph correspond to corners of the polygon. Two nodes are linked by an edge iff there is a maximal rectangle that covers the two corresponding corners. Since a maximal rectangle can cover more than two corners, the reformulation as a graph problem is not yet valid. The following lemma allows us to consider only rectangles that cover at most two corners.

Lemma 7: If three or more corners are covered by a maximal rectangle R, then one of those corners can be covered only by R.

Thus any rectangle that covers three or more corners must belong to any MC of the corners. We denote the set of such rectangles R_3, and delete the corners covered by them from the node set of the graph. Now we are left with a set of nodes in which at most two can be linked by a single edge, giving a valid graph. It is known that a maximum matching in a graph can be found in time $O(|V|^{2.5})$ [7]. The edges in the matching correspond to rectangles that cover two corners; call this set R_M. Any nodes not saturated by the matching correspond to corners that require their own rectangle in the MC; call this set R_U. We now have a MC of the corners:

$$C_{\min} = R_3 \cup R_M \cup R_U .$$

The MC can be found in this manner regardless of whether or not the polygon is in general position. The algorithm has time complexity $O(n^{2.5})$. See [6].

5. COVERING GENERAL POSITION ORTHOGONAL POLYGONS

In this section the effect of restricting the OP to be in general position is considered. The covering problems for the interior of an OP, for the perimeter segments, and for the notches remain NP-complete. The problem of finding a MC of only the horizontal perimeter segments can be solved in polynomial time by an algorithm based on finding a maximum matching in a bipartite graph. This leads to an approximation algorithm for the NP-complete problem of finding a MC of the perimeter segments of a general position OP.

5.1 General Position Interior Cover

If all of the component figures used in Section 2 were in general position, then the OP constructed to simulate an arbitrary Boolean expression would also be in general position, since the definition of general position only forbids *internal* collinearity of perimeter segments. The 'not', 'and/or', 'crossover', and 'switchback' figures are not in general position, but they can easily be modified to satisfy the definition. The modifications consist of the addition of some tab-shaped areas where necessary to eliminate internal collinearities. Each of these tab-shaped areas requires a unique rectangle (called *support edge rectangles* in [2]) in a MC, and has no effect on the operation of the component figures. The modified component figures are shown in [6]. Using these modified component figures, we can state replacements for Lemmas M2, M3, M5, and M6.

Lemma M2': A general position 'not' figure can be covered with 7 1/2 rectangles iff the values on the wires connected to it are complementary.

Lemma M3': A general position 'and/or' figure can be covered with 14 rectangles iff the OR output value is the logical OR of the inputs and the AND output value is the logical AND of the inputs.

Lemma M5': A general position 'crossover' figure can be covered with 13 rectangles iff the truth values of the opposing wires are identical.

Lemma M6': A general position 'switchback' figure can be covered with 8 rectangles iff the truth values at both ends are identical.

Using the modified component figures, the algorithm given in [1] constructs, in polynomial time, a general position OP that simulates an arbitrary Boolean expression and its dual. The balance of the NP-completeness proof is completely analogous to that given in [1] and summarized in Section 2. Thus we can state the following corollary to Theorem M1.

Corollary 2: The problem "are there k rectangles that cover an arbitrary general position OP?" is NP-complete.

5.2 General Position Perimeter Covers

The proof of NP-completeness for the general position perimeter cover problem requires the use of the modified 'not', 'crossover', and 'switchback' figures. However, the modified 'and/or' cannot be used for the same reason that the 'and/or' could not be used in Section 2: under one of the four possible input combinations, an ambiguity arises admitting two different MCs of the perimeter segments. Instead of using the 'and/or', the 'double and' and 'double or' figures from Section 2 can be modified to be in general position as shown in Figure 5. Once again, the modifications consist only of the addition of some tab-shaped areas where necessary to eliminate collinearities. Using the modified figures, we can state replacements for Lemmas 2.2, 2.3, 2.5, and 2.6.

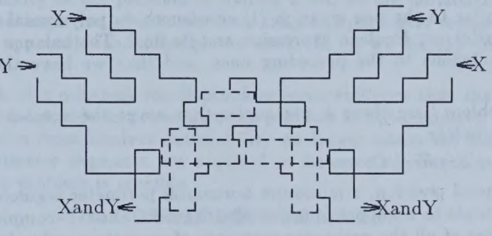


Figure 5

Lemma 2.2': The perimeter segments of a general position 'not' figure can be covered with 7 1/2 rectangles iff the values on the wires connected to it are complementary.

Lemma 2.3': The perimeter segments of a general position 'double and' figure can be covered with 28 rectangles iff the AND output value is the logical AND of the inputs. The perimeter segments of a general position 'double or' figure can be covered with 29 rectangles iff the OR output value is the logical OR of the inputs.

Lemma 2.5': The perimeter segments of a general position 'crossover' figure can be covered with 13 rectangles iff the truth values of the opposing wires are identical.

Lemma 2.6': The perimeter segments of a general position 'switchback' figure can be covered with 8 rectangles iff the truth values at both ends are identical.

The proofs of Lemmas 2.2', 2.5', and 2.6' are a result of the application of Lemma 1 (from Section 3) to Lemmas M2', M5', and M6'. The proof of Lemma 2.3' is based on the fact that the modifications to the 'double and' and 'double or' figures consist only of the addition of some tab-shaped areas where required to remove collinearities. Each of the added areas requires a unique support edge rectangle in any MC and has no effect on the operation of the figures.

With the addition of the general position 'double and' and 'double or' to the set of component figures, an algorithm similar to the one given in [1] constructs, in polynomial time, a general position OP that simulates an arbitrary Boolean expression and its dual. The balance of the NP-completeness proof is completely analogous to that given in [1] and summarized in Section 2. Thus we can state the following corollary to Theorem 1.

Corollary 3: The problem "are there k rectangles that cover the perimeter of an arbitrary general position OP?" is NP-complete.

5.3 General Position Notch Covers

The proof of NP-completeness for the general position notch cover problem uses the same component figures that were used for the general position perimeter cover problem. We can apply Lemma 6 to the general position 'double and', 'double or', 'crossover', and 'switchback' figures to obtain replacements for Lemmas 5.3, 5.5, and 5.6.

Lemma 5.3': The notches of a general position 'double and' figure can be covered with 28 rectangles iff the AND output value is the logical AND of the inputs. The notches of a general position 'double or' figure can be covered with 29 rectangles iff the OR output value is the logical OR of the inputs.

Lemma 5.5': The notches of a general position 'crossover' figure can be covered with 13 rectangles iff the truth values of the opposing wires are identical.

Lemma 5.6': The notches of a general position 'switchback' figure can be covered with 8 rectangles iff the truth values at both ends are identical.

We cannot apply Lemma 6 to the general position 'not' component figure, but it has three MCs, each with the desired Boolean attributes, just as the original 'not' figure did.

Lemma 5.2': The notches of a general position 'not' figure can be covered with 7 1/2 rectangles iff the truth values on the wires connected to it are complementary.

The 'wire', 'generator', 'true', and 'false' figures are already in general position so we can apply Lemma 6 to Lemmas 2.1, 2.4, and 2.7:

Lemma 5.1': The notches of a length k 'wire' figure can be covered with k rectangles iff its input and output values are identical. (Length of wire = number of bends in figure.)

Lemma 5.4': The notches of a 'generator' figure can be covered with 11 rectangles iff all of the wires connected to the generator have the same value.

Lemma 5.7': The notches of a 'true' ('false') figure can be covered with 4 (3 1/2) rectangles iff the input to the figure is TRUE (FALSE).

An algorithm similar to the one given in [1] constructs, in polynomial time, a general position OP that simulates an arbitrary Boolean expression and its dual. The balance of the NP-completeness proof is completely analogous to the preceding ones, and thus we have the following corollary to Theorem 2.

Corollary 4: The problem "are there k rectangles that cover the notches of an arbitrary general position OP?" is NP-complete.

5.4 Horizontal Perimeter Segment Covers

For an OP in general position, a minimum horizontal perimeter segment cover can be found in polynomial time. This leads to an approximation algorithm for the NP-complete problem of finding a minimum rectangle cover of all the perimeter segments of an orthogonal polygon in general position. For an OP not in general position, the complexity of finding a minimum horizontal perimeter segment cover remains unknown.

5.4.1 General Position Horizontal Perimeter Segment Covers. When the OP is in general position, a minimum rectangle cover of the horizontal perimeter segments can be found by reformulating the problem into one of finding a maximum matching in a bipartite graph. Given an OP, we associate a bipartite graph $G=(V,E)$ with it as follows.

Each horizontal perimeter segment is of one of two types, top or bottom. The node set of the graph consists of horizontal perimeter segments, with the natural partition into two sets of type top and bottom, $V = T \cup B$.

A maximal rectangle *covers* a perimeter segment iff the segment is a subset of the perimeter of the rectangle. The edge set of G consists of maximal rectangles that cover two horizontal perimeter segments,

$$E = \{ \textit{maximal rectangles that cover two horizontal perimeter segments} \}.$$

Since the OP is in general position, a maximal rectangle can cover at most one perimeter segment of type top and one of type bottom. Thus in the corresponding graph, an edge is incident on at most two nodes, one from the set T and one from the set B, and the graph is bipartite. The graph may have some isolated nodes.

The graph as we have defined it reflects information only about those rectangles that completely cover at least one horizontal perimeter segment. If it were possible that in a MC some horizontal perimeter segment required more than one rectangle to cover it, then our reformulation of the problem would be inadequate to discover this. The following lemmas preclude this possibility.

Lemma 8: If a horizontal perimeter segment is covered by a collection of maximal rectangles, then it is covered by some single rectangle in that collection.

Lemma 9: In any MC of the horizontal perimeter segments, every rectangle (completely) covers one or two horizontal perimeter segments.

As a result of Lemma 9, when searching for a MC we need only consider those rectangles that (completely) cover at least one horizontal perimeter segment. Thus every horizontal perimeter segment falls into one of two classes: those that have the potential to be matched with others of opposite type (this potential being represented by an edge in our graph), and those requiring their own maximal rectangles (isolated nodes in the graph).

It is known that a maximum matching of a bipartite graph can be found in $O(|V|^{1/2}|E|)$ time [7]. Let M be a maximum matching for graph G, and let U be the set of vertices not saturated by M. Each unsaturated vertex represents a horizontal perimeter segment that cannot be matched with any other segment, and thus requires a corresponding rectangle in the MC. Let R_U be the set of rectangles required to cover these segments, with $|R_U| = |U|$. Each edge in the matching M corresponds to a rectangle that covers two perimeter segments. Thus $M \cup R_U$ is a minimum cover for all the horizontal perimeter segments, call it C_H.

The algorithm for finding a MC of the horizontal perimeter segments of a general position OP has time complexity $O(n^{2.5})$. See [6].

A MC for the vertical perimeter segments, C_V, can be found in a similar manner. Suppose C_M is a MC for all the perimeter segments. Then we have

$$|C_H| \leq |C_M|$$
$$|C_V| \leq |C_M|$$
$$|C_H \cup C_V| \leq 2|C_M|.$$

So $C_H \cup C_V$ is a cover of all the perimeter segments, and has size at most twice that of a MC.

5.4.2 Non-General Position Horizontal Perimeter Segment Covers. When the OP is not in general position, the complexity of the problem of finding a MC of the horizontal perimeter segments is unknown. A maximal rectangle can now cover more than one perimeter segment of a given type, and so the graph matching approach can no longer be used. In [6] we display an example showing that $\theta \neq \alpha$ for this problem.

Although this lack of a min-max result provides some evidence that the covering problem may be "hard," we have not been able to prove that the problem is NP-complete. The component figures used in the transformation from Boolean satisfiability no longer retain the necessary properties when only the horizontal perimeter segments are required to be covered. Perhaps a transformation from some other NP-complete problem is possible.

6. CONCLUDING REMARKS

We feel that the problem just discussed, finding a minimum horizontal perimeter cover when collinearities in the polygon are permitted, isolates one of the key difficulties within the class of rectangle covering problems: when one rectangle can cover more than two items simultaneously, graph matching becomes hypergraph matching, an NP-complete problem in many of its guises.

One of the mysteries in these rectangle covering problems is whether the restriction to a simple polygon (without holes) can lead to polynomial-time algorithms. If this could be established for any of the restricted covering problems studied in this paper, it would represent a major advance in our understanding of geometric covering problems.

ACKNOWLEDGEMENTS

This work was partially supported by NSF grant DCR-83-51468, and grants from Martin-Marietta and IBM.

REFERENCES

[1] W. J. Masek, "Some NP-complete set covering problems," unpub. manuscript, Aug 1979.

[2] S. Chaiken, D. J. Kleitman, M. Saks, and J. Shearer, "Covering regions by rectangles," *SIAM J. Alg. Discrete Methods*, vol. 2, 394-410, Dec 1981.

[3] D. S. Franzblau and D. J. Kleitman, "An algorithm for constructing regions with rectangles: independence and minimum generating sets for collections of intervals," Proc. 16th Annual ACM Symposium on Theory of Computing, 1984, pp. 167-174.

[4] E. Gyŏri, "A minimax theorem on intervals," J. Combinatorial Theory B, 1984, 1-9.

[5] M. R. Garey and D. S. Johnson, *COMPUTERS AND INTRACTABILITY A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company, New York, 1979.

[6] H. E. Conn and J. O'Rourke, "Some restricted rectangle covering problems," Report JHU-87/13, Dept. of Computer Sc., Johns Hopkins University, Baltimore, Jun 1987.

[7] S. Even, *Graph Algorithms,* Computer Science Press, Rockville, MD, 1979.

AN EFFICIENT HEURISTIC ALGORITHM FOR A *3*-SPACE *n*-PARTITIONING PROBLEM USED IN POLYGON RETRIEVAL AND NON-PARAMETRIC STATISTICS

PAWEL GBURZYNSKI
University of Alberta
Department of Computing Science
Edmonton, Alberta, Canada

MARY McLEISH
University of Guelph
Department of Computing and Information Science
Guelph, Ontario, Canada

ABSTRACT

We present an efficient algorithm for solving a certain geometric problem. The problem arose from a very concrete and practical application; however, its nature is much more general. In particular, it is very strongly related to information retrieval problems and non-parametric statistics.

1. INTRODUCTION

1.1 *The Problem Statement*

Assume that we are given two regions of data points R_1 and R_2 in 3-space such that region R_1 contains points with $x>0, z>0$ and region R_2 contains points with $x>0, z<0$ (we assume that $x>0$ for all data points). We want to find a plane passing through the origin which cuts region R_1 into 2 subregions containing the same number of points, and simultaneously partitions region R_2 in a similar manner. The problem has natural analogues in other dimensions.

1.2. *The Main Result*

We present an efficient algorithm for solving the problem. The algorithm consists of several heuristic phases followed by a complete phase. The expected time complexity of the complete phase is $O(N \log\log N)$ (for uniformly distributed data). However, due to the heuristic preprocessing, the observed behaviour of the algorithm is linear all practically interesting data distributions. In particular, for balanced distributions of data points, the proposed algorithm beats the tricky $O(N)$ worst-case solution resulting from the application of the Megiddo technique (cf. [8], [9]). The algorithm can be modified to solve the analogues of the problem in other dimensions.

1.3. *The Significance of the Result*

The problem is strongly related to some techniques in *non-parametric statistics* ([5], [6], [7]) and to the problem of constructing *near-ideal J-way polygon trees* ([2]) which are used in solving several information retrieval problems.

2. PREVIOUS ATTACKS

The time complexity of the natural "brute force" algorithm is $O(N^3)$ which is unacceptable for most applications. For uniformly distributed data, the two-dimensional Willard's algorithm

([2]) performs in $O(N \log N)$ expected time. However, it leaves at least two open problems. First, it isn't clear how to generalise the algorithm for higher dimensions. Second, it seems that by improving the method of selecting the starting parameters (the algorithm consists of a series of iterations starting from some initial approximation of the solution) one can significantly reduce the execution time of the algorithm in a practical implementation.

In [1], several approaches were tried, including some techniques from pattern recognition. However, the best results were obtained using the so-called *median plane* heuristic. On uniform and normally distributed data in 3-space, $O(N)$ expected time behaviour was observed. However, for certain "bad" distributions of data points, the algorithm occasionally exhibited the $O(N^3)$ (worst-case) behaviour.

In [8] Megiddo shows how to solve the two-dimensional version of the problem in the linear worst-case time. The Megiddo's method can be lifted to higher dimensions; however, it is quite tricky and it's practical usefulness is not clear.

3. THE PROPOSED ALGORITHM

The algorithm (which may be viewed as the final link of a stepwise refinement process) is a combination of several heuristic techniques and a substantially modified (and lifted to 3 dimensions) version of the Willard's method.

3.1 *The Structure of the Algorithm*

Assume that n_1 and n_2 denote the numbers of points in R_1 and R_2 respectively. We are looking for a plane $S_{1/2}^{1/2}$ passing through the origin that divides each of the two regions into half.

The algorithm consists of two stages. The first stage is a modified version of *MP* (*median plane* algorithm proposed in [1]). The idea of *MP* (in its original version) was to perform two fast heuristic steps followed by exhaustive search in a limited set of possibilities. The modified version of *MP* consists of three heuristic phases (the exhaustive search phase has been removed). The second stage of the algorithm is executed only if all three heuristic phases of *MP* fail to find a solution. Actually, the second stage is a complete (and efficient) algorithm for solving the partioning problem and it can be used alone. The second stage also consists of three phases. The purpose of the first phase (*SI*) is to determine the starting point for the third (iterative) phase. The second phase (*RR*) attempts to reduce the problem size by eliminating some of the data points so that they can be ignored in phase 3. The third phase (*DS*) performs a series of iterations that generate converging approximations of the solution.

3.2 *Stage 1 (Algorithm MP)*

1. Find the so-called median plane (S_m) uniquely described by two points p_1^m and p_2^m (one for each region) such that $p_1^m = (x_1^m, y_1^m, z_1^m)$ and x_2^m, y_2^m, z_2^m are the medians of x, y, z coordinates of the points in the upper region (R_1). Similarly, point p_2^m is found for R_2. The median plane S_m is fit through the origin, p_1^m and p_2^m.
2. Determine the order in which the three coordinates x, y, z will be processed. Let $Ax + By + Cz = 0$ be the equation of S_m. The order of coordinates is determined by the decreasing order of the absolute values of A, B and C.
3. Put $S = S_m$, $p_1 = p_1^m$ and $p_2 = p_2^m$. Select the first coordinate to be processed.
4. Compute the number of points on both sides of S in R_1. If S already divides R_1 into half, nothing is done for this region. Assume that the number of points on the positive side of S (with respect to the selected coordinate) is greater than the number of points on

the other side (the opposite case is analogous). For each point p_1^i in R_1 located on the positive side of S fit a plane S_1^i passing through p_1^i, p_2 and the origin, and calculate the angle λ_1^i between S_1^i and S. Let k_1 denote the number of points in R_1 located on the positive side of S. Fit a plane S' passing through the origin, p_2 and the point of R_1 with the $(2k_1-n_1)/2+1$'st smallest value of λ_1^i. In reality, if n_1 is even, the plane is shifted slightly (in the negative direction of the selected coordinate) - to make sure that each point of R_1 lies definitely on one of its sides. (Note that S' divides R_1 into half.) Next, adjust the selected coordinate of p_1 as to make that point belong to S' and put $S=S'$.

5. If S divides R_2 into half, the solution is found. Otherwise, perform step 4 for R_2.
6. Iterate steps 4 and 5 until a solution is found or a stall situation is reached. By a stall situation we mean a situation when a full iteration of steps 4 and 5 fails to improve the partitioning of the two regions. In such a case put $S=S_m$, $p_1=p_1^m$ and $p_2=p_2^m$ (start from the original median plane). Select the second coordinate.
7. Perform steps 4, 5, 6 for the selected coordinate. If a stall situation occurs, put $S=S_m$, $p_1=p_1^m$ and $p_2=p_2^m$. Select the last coordinate.
8. Perform steps 4, 5, 6 for the last coordinate.

Each of the three phases of the algorithm presented above consists of a series of iterations, each iteration attempting to produce a better approximation of the solution. However, as the two regions are examined independently we may eventually get into a stall situation (return to a plane considered previously). Therefore, stage 1 may fail to find a solution in which case the algorithm goes to stage 2.

3.3. *Stage 2*

For methodological reasons, it is better to present the phases of stage 2 starting from phase 3 (the proper part of the algorithm).

Note that any solution to the problem $S_{1/2}^{1/2}$ must cross plane $z=0$ along a straight line s_0 passing through the origin. Assume that we can restrict ourselves to some angular interval on the plane $z=0$ (determined by two straight lines l and r passing through the origin) and only consider such potential solutions for which the corresponding intersection lines with plane $z=0$ lie within that interval. Under the above assumption, the following algorithm may be used to solve the problem:

Algorithm DS (phase 3)

1. Find a straight line s on plane $z=0$ dividing the angle between r and l into half.
2. Determine W_1 - the set of planes passing through the origin and intersecting the plane $z=0$ along s, such that each plane from W_1 divides the upper region R_1 into half. If the number of points in R_1 is odd, we assume that "half" means "half up to one point". Note that W_1 is described by the angle between two planes intersecting along s.
3. Similarly, find set W_2 for R_2.
4. If W_1 and W_2 (viewed as sets of planes) have a non-empty intersection, any member of that intersection is a solution to the problem. Otherwise, depending on the relative position of W_1 and W_2 (viewed as angles between pairs of planes), put either $r=s$ or $l=s$; then go back to 1.

It can be shown that if the points in each region are generated by a uniform distribution on any convex and bounded area in the corresponding part of the three-dimensional space (and if the initial angular interval $[l, r]$ is selected in a "reasonable" way) then the expected time of the algorithm presented above is $O(N \log N)$, where N is the total number of points in both regions.

The expected behaviour of *DS* can be improved if the bisection is replaced by an interpolation technique. Let U_1^1 and U_1^2 be the two planes delimiting W_1. Similar planes (U_2^1 and U_2^2) are defined for the second angular set. Let θ_1^1 denote the angle between U_1^1 and plane z=0, and θ_1^2 denote the corresponding angle for U_1^2. Similar angles θ_2^1 and θ_2^2 are defined for U_2^1 and U_2^2. We introduce the following measure of the *distance* between W_1 and W_2:

$$d\,(W_1,W_2) = \text{abs}\left\{\frac{(\cos\theta_1^1+\cos\theta_1^2-\cos\theta_2^1-\cos\theta_2^2)}{2}\right\}.$$

It can be easily seen that d is as a measure of *displacement* between the two angular sets and thus, it can be used as an interpolation function. This idea results in an algorithm with $O(N \log\log N)$ expected time complexity (cf. [4]).

The following algorithm is used to select the initial angular interval $[l, r]$:

Algorithm SI (Phase 1)

1. For R_1 (the upper region), find two horizontal planes H_1^l and H_1^u parallel to the y-axis, passing through the origin and delimiting that region vertically.
2. Similarly, find planes H_2^l and H_2^u for R_2.
3. Find a vertical plane $V_1^{1/2}$ (parallel to the z-axis) passing through the origin that divides R_1 into half.
4. Similarly, find plane $V_2^{1/2}$ for R_2.
5. Let c_1^u and c_1^l denote two straight lines (passing through the origin) resulting from intersection of $V_1^{1/2}$ with H_1^u and H_1^l respectively. Let c_2^u and c_2^l denote corresponding lines for R_2. Fit two planes Z_l - containing c_1^l and c_2^l, and Z_u - containing c_1^u and c_2^u.
6. Let s_l and s_u denote two straight lines intercepted on plane z=0 by planes Z_l and Z_u respectively. Depending on the mutual location of R_1 and R_2 one of the two lines lies to the left of the other one. If s_l lies to the left of s_u, put $l = s_l$, $r = s_u$; otherwise, put $l = s_u$, $r = s_l$. Lines l and r describe the initial angular interval on plane z=0.

The angular interval $[l, r]$ determined by *SI* is selected correctly. In particular, the following lemma holds:

Lemma 1

If no three points from R_1 and R_2 lie on a common plane passing through the origin then there exists a solution plane $S_{1/2}^{1/2}$ crossing $V_1^{1/2}$ between c_1^l and c_1^u, and crossing $V_2^{1/2}$ between c_2^l and c_2^u.

Proof

Without loss of generality we can restrict ourselves to the upper region (R_1). Assume that a solution plane S doesn't cross plane $V_1^{1/2}$ between c_1^l and c_1^u. We can assume that S crosses $V_1^{1/2}$ below c_1^l. The other case is symmetric. Because $V_1^{1/2}$ divides R_1 into half, the part of R_1 delimited by S and $V_1^{1/2}$ must be empty. Let us add two extra points p_0 and p_1 to

R_1. The first point (p_0) may be any point on plane Z_l that fits into the area of R_1 delimited by S, $V_1^{1/2}$ and H_1^u. Actually, we may have to shift p_0 slightly to make sure that the assumption of the lemma is fulfilled. The second point (p_1) is put into the other part of R_1 - sufficiently far to make its location immaterial. Now, the solution plane S_1 (for $R_1 \cup \{p_0, p_1\}$) must pass above c_1^l. By removing the two additional points from R_1 we obtain a solution plane S_1 for the original region passing above c_1^l. Exactly the same method allows us to handle the case when S passes above c_1^l.

■

Besides the fact that the method presented above produces very narrow initial angular intervals for DS, it can be extended to reduce the sizes of R_1 and R_2 by eliminating some of the points, so that they can be ignored in subsequent iterations of DS.

Algorithm RR (Phase 2 - a continuation of SI)

1. If $V_1^{1/2}$ is located to the left of $V_2^{1/2}$, eliminate from R_1 all the points that lie to the right of Z_u. If $V_1^{1/2}$ is located to the right of $V_2^{1/2}$, eliminate from R_1 all the points that lie to the left of Z_l. In the following steps we will assume the first case (i.e. $V_1^{1/2}$ located to the left of $V_2^{1/2}$). The other case is symmetric. Note that if $V_1^{1/2} = V_2^{1/2}$, we already have a solution.
2. Let d_1 denote the straight line resulting from the intersection of Z_u with H_1^l. Find a plane Y_1 passing through the origin and containing d_1, that divides R_1 into half and crosses plane $V_1^{1/2}$ between c_1^l and c_1^u. Note that the existence of such a plane is a corollary from lemma 1.
3. Let e_1 denote the line of intersection of Y_1 with H_1^u. Fit a plane T_1 containing e_1 and c_1^l.
4. Remove from R_1 all the points located to the left of T_1.
5. Perform steps 1-4 for R_2. Let k_1 and k_2 denote the numbers of points removed from the left (with respect to $V_1^{1/2}$ and $V_2^{1/2}$) parts of R_1 and R_2 respectively. Before proceeding with DS_b modify the problem as to divide the two regions (whatever remains of them) in such a way that the solution plane (let us denote it by S_r) leaves $n_1/2 - k_1$ points of R_1 and $n_2/2 - k_2$ points of R_2 on its left side. Note that a trivial modification of DS can enable it to solve such problems.

Depending on the mutual location of R_1 and R_2, the algorithm presented above can eliminate (in $O(N)$ time) a significant part of the total number of points, and in consequence, substantially reduce the problem size for DS. To show that the reduction performed by RR is legitimate the following fact can be proved:

Lemma 2

The plane S_r found by DS_b for the regions reduced by RR is a correct solution for the original problem.

Proof

Let us assume that $V_1^{1/2}$ is located to the left of $V_1^{1/2}$. Moreover, we will restrict ourselves to R_1. The other case, as well as the reduction of the second region, is exactly symmetric.

Lemma 2 allows us to consider only such potential solutions S that cross planes $V_1^{1/2}$ and $V_2^{1/2}$ between pairs of lines c_1^l, c_1^u and c_2^l, c_2^u respectively. Therefore, we can ignore the points of R_1 lying to the right of Z^u as no plane S will ever cross that area. By repeating the arguments used in the proof of lemma 2 we can show that it is enough to consider only such planes S that (besides crossing $V_1^{1/2}$ and $V_2^{1/2}$ within the corresponding intervals) cross the plane Y_1 between d_1 and e_1. This follows from the fact that Y_1 divides R_1 into half. Any plane S satisfying those two conditions cannot cross the area of R_1 located to the left of T_1.

■

4. EXPERIMENTAL RESULTS

The performance of *DS* combined with *MP*, *SI* and *RR* has been analysed experimentally and compared to the performance of the Megiddo's method (cf. [8], [9]). The algorithms were programmed in *C* (using similar data structures and programming techniques) and run on a VAX-11/780. It is important to notice that the Megiddo's method is much more difficult to program and that it uses over three times more memory than the heuristic algorithm proposed in this paper. Data samples were generated according to several types of distribution of points including uniform, normal, Zipf (cf. [3]) and "bad" distributions.

The data sets generated according to the "bad" distribution were intentionally malicious for *MP* and *DS*. The points in each region were grouped into clusters. The clusters and their diameters were uniformly distributed. The cardinalities of the clusters were distributed as consecutive powers of two.

Figures 1, 2, 3, 4 compare the performance of *DS* (with and without *MP*) to the performance of the Megiddo's method (denoted by *MG*). Note that for the uniform distribution of data points *MP* was 100% successful (see table 1).

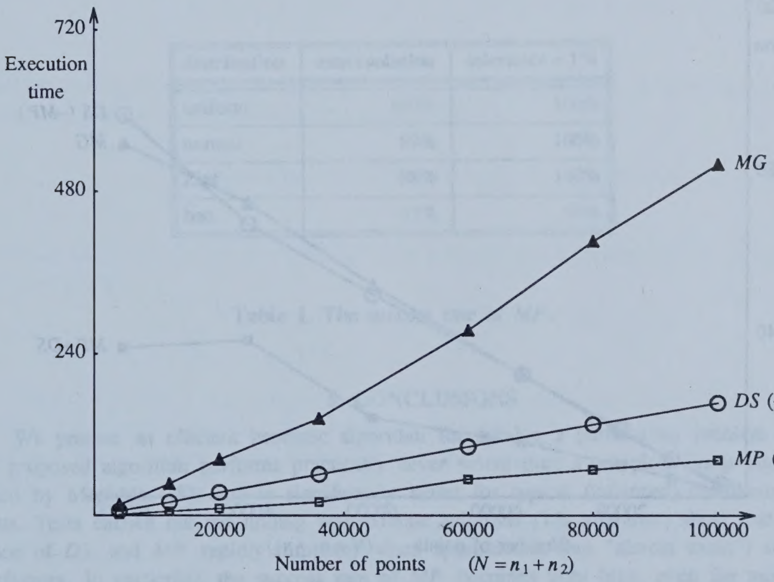


Figure 1. Execution time of *DS* and *MG* (uniform distribution).

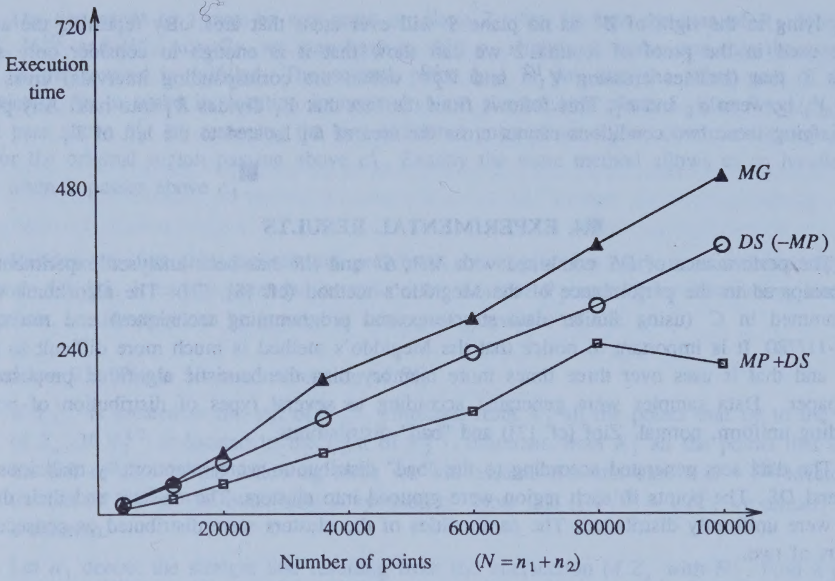


Figure 2. Execution time of *DS* and *MG* (Zipf distribution).

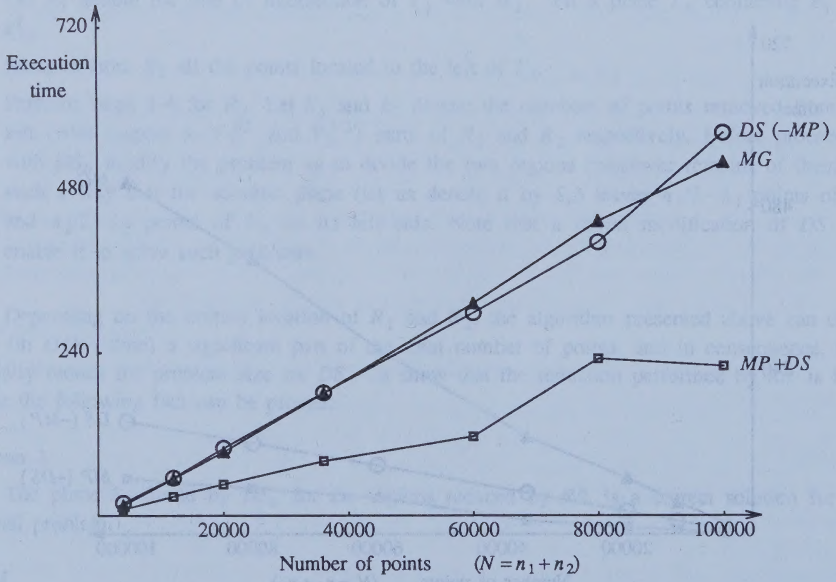


Figure 3. Execution time of *DS* and *MG* (normal distribution).

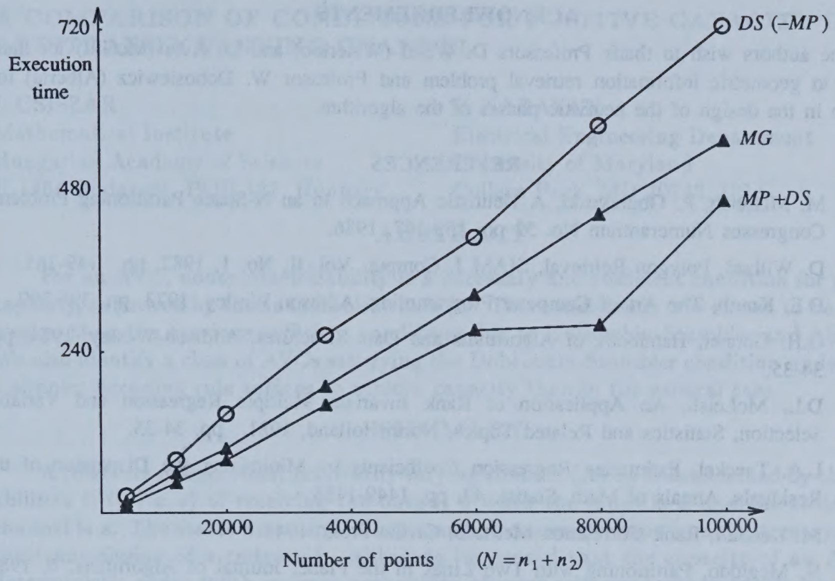


Figure 4. Execution time of *DS* and *MG* ("bad" distribution).

distribution	exact solution	tolerance = 1%
uniform	100%	100%
normal	99%	100%
Zipf	88%	100%
bad	45%	96%

Table 1. The success rate of *MP*.

5. CONCLUSIONS

We present an efficient heuristic algorithm for solving a partitioning problem in 3-space. The proposed algorithm performs practically never worse than a complete linear algorithm suggested by Megiddo ([8]) and is significantly better for typical (balanced) distributions of data points. Tests carried out for finding approximate solutions (1% tolerance) show that the performance of *DS* and *MP* rapidly improves when approximate (but "almost exact") solutions are satisfactory. In particular, the success rate of *MP* becomes very high, even for malicious data samples (see table 1). The Megiddo's method does not exhibit a similar behaviour. The proposed algorithm is relatively easy to program, requires very little additional memory and can be naturally generalised to solve higher dimensional analogues of the partitioning problem.

ACKNOWLEDGEMENTS

The authors wish to thank Professors D. Wood (Waterloo) and D. Avis (McGill) for their referral to geometric information retrieval problem and Professor W. Dobosiewicz (Alberta) for his help in the design of the heuristic phases of the algorithm.

REFERENCES

[1] M. McLeish, P. Gburzynski, A Heuristic Approach to an N-Space Partitioning Problem, Congressus Numerantium No. 52 pp. 153-167, 1986.

[2] D. Willard, Polygon Retrieval, SIAM J. Comput. Vol. II, No. 1, 1982, pp. 149-165.

[3] D.E. Knuth, The Art of Computer Programming, Addison-Wesley, 1973, pp. 396-399.

[4] G.H. Gonnet, Handbook of Algorithms and Data Structures, Addison-Wesley, 1984, pp. 34-35.

[5] D.L. McLeish, An Application of Rank Invariant Multiple Regression and Variable Selection, Statistics and Related Topics, North-Holland, 1981. pp. 34-35.

[6] L.A. Taeckel, Estimating Regression Coefficients by Minimising the Dispersion of the Residuals, Annals of Math. Statist. 43, pp. 1449-1458.

[7] M. Kendall, Rank Correlation Methods, Griffin Press, 1948.

[8] N. Megiddo, Partitioning with Two Lines in the Plane, Journal of Algorithms, 6, 1985, pp. 430-433.

[9] N. Megiddo, Linear Time Algorithms for Linear Programming in R^3 and Related Problems, SIAM J. Comput. 12, 1983. pp. 759-776.

A COMPARISON OF CONDITIONS FOR POSITIVE CAPACITY OF AN ARBITRARILY VARYING CHANNEL

I. CSISZÁR
Mathematical Institute
Hungarian Academy of Sciences
H-1364 Budapest, POB 127, Hungary

P. NARAYAN
Electrical Engineering Department
University of Maryland
College Park, MD 20742, USA

ABSTRACT

For an AVC, nonsymmetrizability is a necessary and sufficient condition for positive capacity, as proved by the authors elsewhere [2]. This condition is now shown to be strictly weaker than the previous sufficient conditions due to Dobrushin-Stambler and Ahlswede. We also identify a class of AVCs satisfying the Dobrushin-Stambler condition under which a simpler decoding rule suffices to achieve capacity than in the general case.

SUMMARY

A (discrete memoryless) arbitrarily varying channel (AVC) is determined by the probabilities $W(y \mid x, s)$ of receiving the output y when the input is x and the state of the channel is s. The states may vary in any manner, chosen perhaps by an adversary, during the transmission of a codeword. Ahlswede [1] proved that the capacity of an AVC for deterministic codes (with average probability of error criterion) either equals its random code capacity or else is zero. His sufficient condition for positive capacity is that for a suitable pair of input distributions P_1, P_2, no distributions Q_1, Q_2 on the states can yield

$$\sum_{x,s} P_1(x)W(y \mid x,s)Q_1(s) = \sum_{x,s} P_2(x)W(y \mid x,s)Q_2(s). \tag{1}$$

Another sufficient condition, due to Dobrushin-Stambler [3] is that for a suitable P, no stochastic matrix U and distribution Q exist with

$$\sum_{x,s} P(x)W(y \mid x,s)U(s \mid x') = \sum_{s} W(y \mid x',s)Q(s). \tag{2}$$

The necessary and sufficient condition [2] is nonsymmetrizability, i.e., the nonexistence of a U rendering

$$\sum_{s} W(y \mid x,s)U(s \mid x') = \sum_{s} W(y \mid x',s)U(s \mid x). \tag{3}$$

As (3)⇒(2)⇒(1), Ahlswede's condition implies Dobrushin-Stambler's, which in turn implies nonsymmetrizability. The following example shows that the Dobrushin-Stambler condition (and, consequently, also Ahlswede's) is strictly stronger than nonsymmetrizability.

Example: Let an AVC with input alphabet $\{0,1,2\}$ and state and output alphabets $\{0,1\}$ be determined by

$$W(0 \mid 0,0) = 1 \quad W(0 \mid 1,0) = 0.9 \quad W(0 \mid 2,0) = 0.8$$
$$W(0 \mid 0,1) = 1 \quad W(0 \mid 1,1) = 0.2 \quad W(0 \mid 1,2) = 0.1.$$

Setting $Q = (1/2, 1/2)$, the equations (2) can be explicitly solved for U. On the other hand, the equations (3) have no admissible solution.

Conditions analogous to the ones above can also be given for positive capacity under a state constraint, when the state sequences $(s_1, \cdots, s_n)$ during the transmission of a codeword must satisfy

$$\frac{1}{n}\sum_{i=1}^{n} \ell(s_i) \le \Lambda, \tag{4}$$

for some given function ℓ and constant Λ. In particular, the analog of Dobrushin-Stambler's condition is that for some P, no U and Q exist that simultaneously satisfy (2) and

$$\sum_{x,s} P(x)U(s \mid x)\ell(s) \le \Lambda, \qquad \sum_{s} Q(s)\ell(s) \le \Lambda. \tag{5}$$

This condition is not necessary for positive capacity. However, if it is satisfied, codes with codewords of type P afford simpler decoding than in general (compare [3] with [2]). Hence, the following theorem is of interest.

Theorem: Consider a deterministic AVC whose output is the sum of its input and state, with the input, state, and output alphabets being (finite) subsets of the nonnegative numbers containing 0. Then, for any increasing function $\ell(\cdot)$ with $\ell(0) = 0$, the capacity under state constraint (4) is positive if and only if $E_P\ell(X) > \Lambda$ for some P. For every such P, the Dobrushin-Stambler condition is satisfied. Moreover, the statements above remain true also in the presence of additive noise.

A similar theorem holds for subsets of the entire real line provided that $\ell(\cdot)$ is a convex function such that $\sum_x P(x)\ell(x+t)$ is minimized for $t = 0$. In the most interesting case, $\ell(s) = s^2$, the Dobrushin-Stambler condition is satisfied for every P with mean 0 and variance $> \Lambda$.

REFERENCES

1. R. Ahlswede, "Elimination of Correlation in Random Codes for Arbitrarily Varying Channels," *Z. Wahrscheinlichkeitstheorie Verw. Gebiete*, vol. 44, pp. 159-175, 1978.
2. I. Csiszár and P. Narayan, "The Capacity of the Arbitrarily Varying Channel Revisited: Positivity, Constraints," *IEEE Transactions on Information Theory*, to appear.
3. R.L. Dobrushin and S.Z. Stambler, "Coding Theorems for Classes of Arbitrarily Varying Discrete Memoryless Channels," *Problemy Peredachi Informatsii*, vol. 11, no. 2, pp. 3-22, 1975 (English translation).

MULTITERMINAL SOURCE ENCODING WITH ONE DISTORTION CRITERION

TOBY BERGER AND RAYMOND YEUNG
Department of Electrical Engineering
Cornell University
Ithaca, NY 14853

ABSTRACT

We investigate the problem of encoding two correlated i.i.d. sources $\{X_k\}$, $\{Y_k\}$ by two separate encoders. Decoding is done by a single decoder which receives the outputs from both encoders. The reconstruction of $\{X_k\}$ is required to be perfect in the usual Shannon sense. We determine the admissible rate region $R(D)$, where D is the distortion of $\{\hat{Y}_k\}$. When $D \geq D_{\max} = \min_{(\hat{y}\in\hat{\mathbf{Y}})} E\, d_y(Y, \hat{y})$ our result reduces to side information theory. When $R_x = H(X)$, it reduces to the Wyner-Ziv result.

INTRODUCTION

Our results build on earlier work of Slepian and Wolf [1], Wyner [2], Ahlswede and Körner [3], Wyner and Ziv [4], Berger et al. [5], and Kaspi and Berger [6]. The communication system we consider is illustrtated in Figure 1.

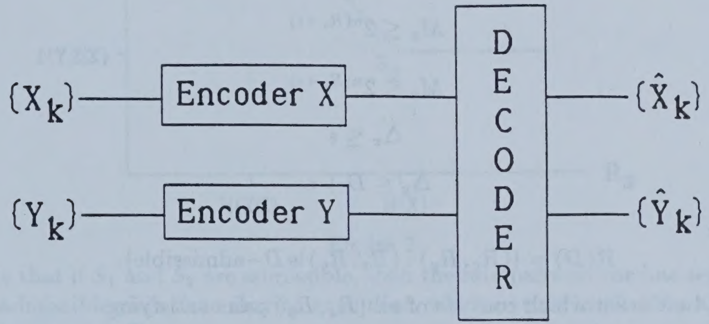


Figure 1.

$\{X_k\}$ and $\{Y_k\}$ are two correlated discrete i.i.d. sources. It is assumed that Encoder X and Encoder Y do not communicate. $\{X_k\}$ and $\{Y_k\}$ are reconstructed as $\{\hat{X}_k\}$ and $\{\hat{Y}_k\}$, respectively, by a decoder that receives the output from both encoders.

We consider the situation in which the reconstruction of $\{X_k\}$ must be perfect in the usual Shannon sense, while that of $\{Y_r\}$ is subject to a distortion criterion. We determine the admissible rate region $R(D)$, where D is the distortion constraint on $\{\hat{Y}_k\}$.

PROBLEM STATEMENT AND MAIN RESULT

Let X and Y be discrete i.i.d. r.v's with joint probability distribution $p(x, y)$; X and Y take values in $\mathbf{X}$ and $\mathbf{Y}$, respectively. Let $\hat{X}$ and $\hat{Y}$ be reconstructions of X and Y, taking values in alphabets $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$, respectively, and let

$$d_x : \mathbf{X} \times \hat{\mathbf{X}} \to \{0, 1\}; \quad d_x(x, \hat{x}) = 1 - \delta_{x\hat{x}}$$

$$d_y : \mathbf{Y} \times \hat{\mathbf{Y}} \to [0, \infty)$$

be distortion measures. An $(n, M_x, M_y, \Delta_x, \Delta_y)$ code is defined by

$$f_x : \mathbf{X}^n \to \{0, 1, \ldots, M_x - 1\}$$

$$f_y : \mathbf{Y}^n \to \{0, 1, \ldots, M_y - 1\}$$

$$g : \{0, 1, \ldots, M_x - 1\} \times \{0, 1, \ldots, M_y - 1\} \to \mathbf{X}^n \times \hat{\mathbf{Y}}^n$$

and

$$E \frac{1}{n} \sum_{k=1}^{n} d_x(X_k, \hat{X}_k) = \Delta_x$$

$$E \frac{1}{n} \sum_{k=1}^{n} d_y(Y_k, \hat{Y}_k) = \Delta_y,$$

where

$$(\hat{\underline{X}}^n, \hat{\underline{Y}}^n) = g(f_x(\underline{X}^n), f_y(\underline{Y}^n)).$$

A rate pair (R_x, R_y) is D-admissible if for any $\epsilon > 0$, there exists for sufficiently large n an $(n, M_x, M_y, \Delta_x, \Delta_y)$ code such that

$$M_x \le 2^{n(R_x + \epsilon)}$$

$$M_y \le 2^{n(R_y + \epsilon)}$$

$$\Delta_x \le \epsilon$$

$$\Delta_y \le D + \epsilon.$$

Define

$$\mathbf{R}(D) = \{(R_x, R_y) : (R_x, R_y) \text{ is } D\text{-admissible}\}.$$

Let $\mathbf{R}^*(D)$ be the set which consists of all (R_x, R_y) pairs satisfying

(i) there exists an auxiliary random variable Z such that $X -o- Y -o- Z$ forms a Markov chain;

(ii) there exists $\hat{Y}(X, Z)$ such that $E\, d_y(Y, \hat{Y}) \le D$.

(iii) $R_x \ge H(X|Z)$; $R_y \ge I(Y; Z|X)$; $R_x + R_y \ge H(X) + I(Y; Z|X)$;

(iv) $|\mathbf{Z}| \le |\mathbf{Y}| + 2$. (**)

Theorem 1: $\mathbf{R}(D) = \mathbf{R}^*(D)$.

The proof of Theorem 1 is divided into two parts. The admissibility of $\mathbf{R}^*(D)$, namely $\mathbf{R}^*(D) \subset \mathbf{R}(D)$, is proved in Section 3. The converse, $\mathbf{R}(D) \subset \mathbf{R}^*(D)$, is proved in Section 4.

Three important special cases of Theorem 1 are

(i) Slepian-Wolf theory [1]: $D = 0$, $d_y(y, \hat{y}) = 1 - \delta_{y\hat{y}}$;
(ii) Side information theory ([2] or [3]): $D \ge D_{\max}$;
(iii) Wyner-Ziv theory [4]: $R_x = H(X)$.

The reductions to these special cases are presented in Section 5.

The situation we consider is a special case of the more general situation studied by Berger et al. [5], in which distortion criteria are imposed on the reconstruction of both sources. They have determined inner and outer bounds for the coding rate region.

Theorem 1 is a simplified form of a special case of the result of Kaspi and Berger [6]. However, the associated reduction of their general result takes considerably more work than does the direct proof of Theorem 1 that we present below. We believe that Theorem 1 is of independent interest. Moreover, we have proved a more general coding theorem [8] which is directly inspired by Theorem 1.

ADMISSIBILITY OF $R^*(D)$

In the definition of $\mathbf{R}^*(D)$, for any Z satisfying (i) and (ii), let $\mathbf{R}_z$ denote the subset of $\mathbf{R}^*(D)$ specified by (iii). This is illustrated in Figure 2.

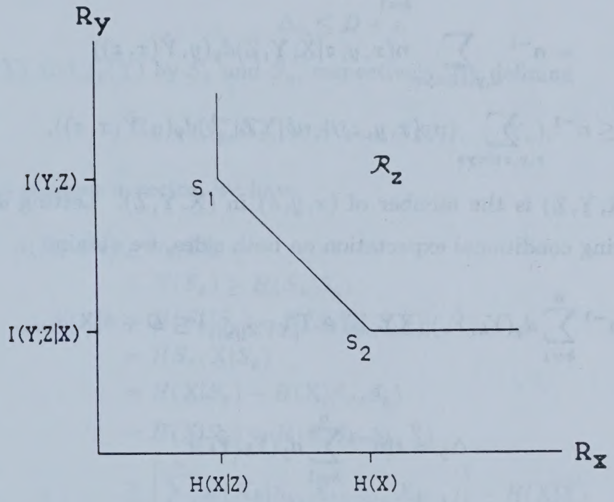


Figure 2

Note that if S_1 and S_2 are admissible, then the rate pairs on the line segment S_1S_2 are also admissible via a time-sharing argument, which in turn implies the admissibility of the whole region $\mathbf{R}_z$.

The admissibility of S_2 is a consequence of Wyner-Ziv theory [4]. The admissibility of S_1 is proved as follows via an application of the so-called side information result ([2] or [3]), which asserts that, for any δ, ϵ, $\epsilon' > 0$ and sufficiently large n, there exists a code for which

$$M_x \leq 2^{n[H(X|Z)+\epsilon]}$$

$$M_y \leq 2^{n[I(Y;Z)+\epsilon]}$$

and by means of which the decoder can reconstruct random n-vectors $\hat{\underline{X}}$ and $\underline{Z}$ such that

$$\Delta_x \leq \epsilon$$

and, in the δ-typical notation of Csiszar and Körner [7],

$$Pr[(\hat{\underline{X}},\, \underline{Y},\, \underline{Z}) \notin T^n_{[XYZ]|\mathrm{x}|\delta}] \leq \epsilon'.$$

Using the function $\hat{Y}$ of (ii), the decoder can form $\hat{Y}$ by taking

$$\hat{Y}_k = \hat{Y}(\hat{X}_k, Z_k),\ 1 \le k \le n.$$

Hence, if $(\underline{\hat{X}}, \underline{Y}, \underline{Z}) \in T^n_{[XYZ]|\mathbf{x}|\delta}$, we have

$$n^{-1}\sum_{k=1}^{n} d_y(Y,k\hat{Y}_k)$$

$$= n^{-1}\sum_{k=1}^{n} d_y(Y_k, \hat{Y}(\hat{X}_k, Z_k))$$

$$= n^{-1} \sum_{x,y,z \in \mathbf{xyz}} n(x,y,z|\underline{\hat{X}}, \underline{Y}, \underline{Z}) d_y(y, \hat{Y}(x,z))$$

$$\le n^{-1} \sum_{x,y,z \in \mathbf{xyz}} (np(x,y,z) + n\delta|\mathbf{YZ}|^{-1}) d_y(y, \hat{Y}(x,z)),$$

where $n(x,y,z|\underline{\hat{X}}, \underline{Y}, \underline{Z})$ is the number of (x,y,z) in $(\underline{\hat{X}}, \underline{Y}, \underline{Z})$. Letting $d_{\max} = \max_{y,y'} d_y(y,y')$ and taking conditional expectation on both sides, we obtain

$$E[n^{-1}\sum_{k=1}^{n} d_y(Y_k, \hat{Y}_k)|(\underline{\hat{X}}\underline{Y}, \underline{Z}) \in T^n_{[XYZ]|\mathbf{x}|\delta}] \le D + \delta|\mathbf{X}|d_{\max}.$$

Therefore

$$\Delta_y = e[n^{-1}\sum_{k=1}^{n} d_y(Y_k, \hat{Y}_k)]$$

$$\le (D + \delta|\mathbf{X}|D_{\max}) \cdot 1 + d_{\max}\epsilon'$$

$$= D + d_{\max}(\delta|\mathbf{X}| + \epsilon').$$

Choosing δ and ϵ' small enough that $\delta|\mathbf{X}| + \epsilon' \le \epsilon/d_{\max}$, we have $\Delta_y \le D + \epsilon$, so S_1 is admissible.

THE CONVERSE

Define the set $\tilde{\mathbf{R}}^*(D)$ consisting of all (R_x, R_y) pairs that satisfy the following four conditions:

(i) there exist auxiliary random variables Z, W such that $X -o- Y -o- (Z,W)$ forms a Markov chain;

(ii) W is independent of X and Y;

(iii) $R_x \ge H(X|ZW)$; $R_y \ge I(Y;Z|XW)$; $R_x + R_y \ge H(X) + I(Y;Z|XW)$;

(iv) there exists $\hat{Y}(X,Z,W)$ such that $E\, d_y(Y,\hat{Y}) \le D$.

We first show that $\mathbf{R}^*(D)$ is convex in the sense that $\mathbf{R}^*(D) = \tilde{\mathbf{R}}^*(D)$. By taking W = constant, it is clear that $\mathbf{R}^*(D) \subset \tilde{\mathbf{R}}^*(D)$. Upon defining a new random variable $Z' = (Z,W)$ in (iii) and (iv), it is seen that $\tilde{\mathbf{R}}^*(D) \subset \mathbf{R}^*(D)$. Thus $\mathbf{R}^*(D) = \tilde{\mathbf{R}}^*(D)$, so $\mathbf{R}^*(D)$ is convex.

Before proving the rate constraints, we first state without proof the following vector version of Fano's inequality.

Lemma 1:

$$n^{-1}H(\underline{X}|\hat{\underline{X}}) \leq \Delta_x \log[|\mathbf{X}|-1] \,:\, H_b(\Delta_x),$$

where $H_b(\cdot)$ is the binary entropy function.

Now assume (R_x, R_y) is D-admissible. Then for any $\epsilon > 0$, there exists for sufficiently large n an $(n, M_x, M_y, \Delta_x, \Delta_y)$ code such that

$$M_x \leq 2^{n(R_x+\epsilon)}$$

$$M_y \leq 2^{n(R_y+\epsilon)}$$

$$\Delta_x \leq \epsilon$$

$$\Delta_y \leq D + \epsilon.$$

Denote $f_x(\underline{X})$ and $f_y(\underline{Y})$ by S_x and S_y, respectively. By defining

$$Z_k = (S_y, X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n),$$

where $\underline{Z}$ is a random n-vector, we have

$$\begin{aligned}
n(R_x+\epsilon) &\geq \log M_x \\
&\geq H(S_x) \geq H(S_x|S_y) \\
&= H(S_x|S_y) - H(S_x|\underline{X}, S_y) \\
&= I(S_x; \underline{X}|S_y) \\
&= H(\underline{X}|S_y) - H(\underline{X}|S_x, S_y) \\
&= H(\underline{X}|S_y) = H(\underline{X}|S_x, S_y, \hat{\underline{X}}) \\
&\geq \left[\sum_{k=1}^{n} H(X_k|S_y, X_1, \ldots, X_{k-1})\right] - H(\underline{X}|\hat{\underline{X}}) \\
&\geq \left[\sum_{k=1}^{n} H(X_k|S_y, X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n)\right] - \\
&\qquad n[\Delta_x \log(|\mathbf{X}|-1) + H_b(\Delta_x)] \quad \text{by Lemma 1}
\end{aligned}$$

$$R_x + \epsilon \geq n^{-1}\sum_{k=1}^{n} H(X_k|Z_k) - [\Delta_x \log(|\mathbf{X}|-1) + H_b(\Delta_x)].$$

We also have

$$\begin{aligned}
n(R_y+\epsilon) &\geq \log M_y \\
&\geq H(S_y) = H(S_y|\underline{X}) + I(\underline{X}; S_y) \\
&= H(S_y|\underline{X}) - H(S_y|\underline{X}, \underline{Y}) + I(\underline{X}; S_y) \\
&= I(\underline{Y}; S_y|\underline{X}) + I(\underline{X}; S_y) \\
&= H(\underline{Y}|\underline{X}) - H(\underline{Y}|S_y, \underline{X}) + I(\underline{X}; S_y) \\
&= \left[\sum_{k=1}^{n} H(Y_k|X_k) - H(Y_k|S_y, \underline{X}, Y_1, \ldots, Y_{k-1})\right] + I(\underline{X}; S_y)
\end{aligned}$$

$$\geq \left[\sum_{k=1}^{n} H(Y_k|X_k) - H(Y_k|S_y, \underline{X})\right] + I(\underline{X}; S_y)$$
$$= \left[\sum_{k=1}^{n} H(Y_k|X_k) - H(Y_k|Z_k, X_k)\right] + I(\underline{X}; S_y)$$
$$= \sum_{k=1}^{n} I(Y_k; Z_k|X_k) + I(\underline{X}; S_y) \qquad (*)$$
$$\geq \sum_{k=1}^{n} I(Y_k; Z_k|X_k).$$

Thus

$$R_y + \epsilon \geq n^{-1} \sum_{k=1}^{n} I(Y_k; Z_k|X_k).$$

Toward proving the rate sum constraint, note that

$$\begin{aligned} n(R_x + \epsilon) &\geq \log M_x \\ &\geq H(S_x) = H(S_x) - H(S_x|\underline{X}) \\ &= H(\underline{X}) - H(\underline{X}|S_x) \\ R_x + \epsilon &= H(X) - n^{-1} H(\underline{X}|S_x) \end{aligned}$$

and from (*) that

$$R_y + \epsilon \geq n^{-1} \sum_{k=1}^{n} I(Y_k; Z_k|X_k) + n^{-1} I(\underline{X}; S_y).$$

Therefore

$$R_x + R_y + 2\epsilon \geq H(X) + n^{-1} \sum_{k=1}^{n} I(Y_k; Z_k|X_k) + n^{-1}[I(\underline{X}; S_y) - H(\underline{X}|S_x)].$$

Now

$$I(\underline{X}; S_y) = I(S_x, \underline{X}; S_y) = I(S_x; S_y) + I(\underline{X}; S_y|S_x)$$

and

$$H(\underline{X}|S_x) = H(\underline{X}|S_x, S_y) + I(\underline{X}; S_y|S_x),$$

so

$$\begin{aligned} I(\underline{X}; S_y) - H(\underline{X}|S_x) &= I(S_x; S_y) - H(\underline{X}|S_x, S_y) \\ &= I(S_x; S_y) - H(\underline{X}|S_x, S_y, \hat{\underline{X}}) \\ &\geq I(S_x, S_y) - H(\underline{X}|\hat{\underline{X}}) \\ &\geq I(S_x, S_y) - n[\Delta_x \log(|\mathbf{X}| - 1) + H_b(\Delta_x)] \text{ by Lemma 1} \\ &\geq -n[\Delta_x \log(|\mathbf{X}| - 1) + H_b(\Delta_x)]. \end{aligned}$$

Thus

$$R_x + R_y + 2\epsilon \geq H(X) + n^{-1} \sum_{k=1}^{n} I(Y_k; Z_k|X_k) - [\Delta_x \log(|\mathbf{X}| - 1) + H_b(\Delta_x)].$$

Note that $\hat{Y}_k = \hat{Y}_k(S_x, S_y) = \hat{Y}_k(X_k, Z_k)$. To establish that $X_k -o- Y_k -o- Z_k$ is a Markov chain, we consider

$$\begin{aligned}
I(X_k; Z_k|Y_k) &= I(X_k; S_y, X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n|Y_k) \\
&= H(X_k|Y_k) - H(X_k|Y_k, S_y, X_1, \ldots X_{k-1}, X_{k+1}, \ldots, X_n) \\
&\leq H(X_k|Y_k) - H(X_k|\underline{Y}, S_y, X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n) \\
&= H(X_k|Y_k) - H(X_k|\underline{Y}, X_1, \ldots X_{k-1}, X_{k+1}, \ldots, X_n) \\
&= H(X_k|Y_k) - H(X_k|Y_k) \\
&= 0.
\end{aligned}$$

Thus $I(X_k; Z_k|Y_k) = 0$, indicating that $X_k -o- Y_k -o- Z_k$ is a Markov chain. Upon defining

$$\gamma = \epsilon + [\Delta_x \log(|\mathbf{X}| - 1) + H_b(\Delta_x)].$$

We see by standard convexity argument that $(R_x + \gamma, R_y + \gamma) \in R^*(D + \gamma)$ for all $\gamma > 0$, since γ can be made arbitrarily small by taking ϵ small enough. By closure and convexity of $\mathbf{R}^*(D)$, we conclude that $(R_x, R_y) \in R^*(D)$.

Finally, for determining $\mathbf{R}^*(D)$ it is sufficient to consider $|\mathbf{Z}| \leq |\mathbf{Y}| + 2$. It can be seen by invoking the support lemma [7, p.310] that $\mathbf{Z}$ must have $|\mathbf{Y}| - 1$ letters to ensure preservation of $p(y)$ plus three more letters to preserve the constraints on $H(X|Z)$, $I(Y; Z|X)$ and D. Thus $\mathbf{R}(D) \subset \mathbf{R}^*(D)$.

THREE IMPORTANT SPECIAL CASES

We now show that Theorem 1 reduces to previous results in appropriate special cases.

Case 1: $D = 0$ and $d_y(y, \hat{y}) + 1 - \delta_{y\hat{y}}$

This corresponds to Y having to be reproduced perfectly in the usual Shannon sense. The coding rate region, $\mathbf{R}_{S-W}$, was determined by Slepian and Wolf [1] to be the set of all (R_x, R_y) pairs satisfying

$$R_x \geq H(X|Y);\ R_y \geq H(Y|X);\ R_x + R_y \geq H(X, Y).$$

With the constraints $D = 0$ and $d_y(y, \hat{y}) = 1 - \delta_{y\hat{y}}$ imposed on $\mathbf{R}(D)$, $\mathbf{R}(D)$ becomes $\mathbf{R}^1$, the set of all (R_x, R_y) pairs satisfying

(i) there exists an auxiliary random variable Z such that $X - o - Y - o - Z$ forms a Markov chain;
(ii) $\hat{Y}(X, A) \equiv Y = Y(X, Z)$;
(iii) $R_x \geq H(X|Z)$; $R_y \geq I(Y; Z|X)$; $R_x + R_y \geq H(X) + I(Y; Z|X)$.

We will show that $\mathbf{R}_{S-W} = \mathbf{R}^1$.

By setting $Z = Y$, we see that $\mathbf{R}_{S-W} \subset \mathbf{R}^1$. To show the converse, assume $(R_x, R_y) \in \mathbf{R}^1$. Then

$$\begin{aligned}
R_x \geq H(X|Z) \geq H(X|Y) \quad &\text{by}\, X - o - Y - o - Z, \\
R_y &\geq I(Y; Z|X) \\
&= H(Y|X) - H(Y|XZ) \\
&= H(Y|X) \quad \text{by}\, Y = Y(X, Z),
\end{aligned}$$

and

$$\begin{aligned} R_x + R_y &\geq H(X) + I(Y;Z|X) \\ &= H(X) + H(Y|X) - H(Y|XZ) \\ &= H(X,Y). \end{aligned}$$

Thus $\mathbf{R}^1 \subset \mathbf{R}_{S-W}$.

Case 2: $D = D_{\max}$

This corresponds to the situation in which there is no constraint on the reproduction of Y. The coding rate region $\mathbf{R}_{S-I}$, determined by Wyner [2] and Ahlswede and Körner [3], is the set of all (R_x, R_y) pairs satisfying

(i) there exists an auxiliary random variable Z such that $X - o - Y - o - Z$ forms a Markov chain;

(ii) $R_x \geq H(X|Z)$; $R_y \geq I(Y;Z)$.

With $D = D_{\max}$, (ii) is removed from (**) and $\mathbf{R}(D)$ becomes $\mathbf{R}^2$, which is the set of all (R_x, R_y) satisfying

(i) there exists an auxiliary random variable Z such that $X - o - Y - o - Z$ forms a Markov chain;

(ii) $R_x \geq H(X|Z)$; $R_y \geq I(Y;Z|X)$; $R_x + R_y \geq H(X) + I(Y;Z|X)$.

To see that $\mathbf{R}_{S-I} \subset \mathbf{R}^2$, we assume $(R_x, R_y) \in \mathbf{R}_{S-I}$ and consider

$$\begin{aligned} R_y &\geq I(Y;Z) \\ &= I(XY;Z) \qquad \text{by } X - o - Y - o - Z \\ &= I(X;Z) + I(Y;Z|X) \\ &\geq I(Y;Z|X) \end{aligned}$$

and

$$\begin{aligned} R_x + R_y &\geq H(X|Z) + I(Y;Z) \\ &= H(X) + I(Y;Z|X) \qquad \text{by } X - o - Y - o - Z. \end{aligned}$$

This shows that $(R_x, R_y) \in \mathbf{R}^2$. Thus $\mathbf{R}_{S-I} \subset \mathbf{R}^2$.

To see the converse, partition the region $\mathbf{R}_z$ in $\mathbf{R}^2$ induced by a Z satisfying (i) into the regions (a), (b) and (c) shown in Figure 3 as follows:

(a) $R_x \geq H(X|Z)$; $R_y \geq I(Y;Z)$;
(b) $R_x \geq H(X)$; $I(Y;Z|X) \leq R_y < I(Y;Z)$;
(c) $R_x < H(X)$; $R_y < I(Y;Z)$; $R_x + R_y \geq H(X) + I(Y;Z|X)$.

If (R_x, R_y) in $\mathbf{R}^2$, then for some Z that satisfies (i) and (ii), it is in either region (a), region (b) or region (c). If (R_x, R_y) is in region (a), then it is in $\mathbf{R}_{S-I}$ by definition. If (R_x, R_y) is in region (b) for some Z, then where Z'=constant, we have $X - o - Y - o - Z'$, $R_x \geq H(X) = H(X|Z')$ and $R_y \geq 0 = I(Y;Z')$, so (R_x, R_y) is in $\mathbf{R}_{S-I}$. Finally, if (R_x, R_y) is in region (c), then it is a linear combination of a point in region (a) and a point in region (b), so it is in $\mathbf{R}_{S-I}$ because $\mathbf{R}_{S-I}$ is convex.

Case 3: $R_x = H(X)$

This is the situation in which the decoder can reconstruct X perfectly in the usual Shannon sense solely from the output of Encoder X. The rate-distortion function of R_y, $R_{W-Z}(D)$, determined by Wyner and Ziv in [4], is given by

$$R_{W-Z}(D) = \min_Z I(Y;Z|X)$$

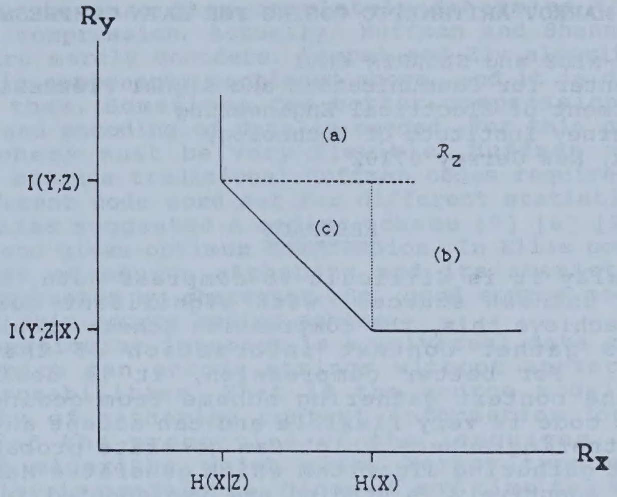


Figure 3.

where Z is an auxiliary random variable satisfying

(i) X –o– Y –o– Z forms a Markov chain;
(ii) there exists $\hat{Y}(X, Z)$ such that $E\, d_y(Y, \hat{Y}) \leq D$.

Upon setting $R_x = H(X)$ in $\mathbf{R}^*(D)$, we see that $R_y \geq I(Y; Z|X)$ for some Z satisfying (i) and (ii) in the specification of $R_{W-Z}(D)$. Thus $R_y \geq R_{W-Z}(D)$, so Theorem 1 reduces to the Wyner-Ziv theory.

REFERENCES

[1] D. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources." *IEEE Transactions on Information Theory*, vol. IT-19, pp. 471-480,

[2] A. D. Wyner, "On Source Coding with Side Information at the Decoder," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 294-300, May 1975.

[3] R. F. Ahlswede and J. Körner, "Source Coding with Side Information and a Converse for Degraded Broadcast Channels," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 629-637, Nov. 1975.

[4] A. D. Wyner and J. Ziv, "The Rate Distortion Function for Source Coding with Side Information at the Decoder," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 1-10, Jan. 1976.

[5] T. Berger, K. Housewright, J. Omura, S-Y. Tund ang J. Wolfowitz, "An Upper Bound on the Rate-Distortion Function for Source Coding with Partial Side Information at the Decoder," *IEEE Transactions on Informatio Theory*, vol. IT-25, pp. 664-668, Nov. 1979.

[6] A. H. Kaspi and T. Berger, "Rate-distortion for Correlated Sources with Partially Separated Encoders," *IEEE Transactions on Information Theory*, vol IT-28, pp. 828-910, Nov. 1982. July 1973.

[7] I.Csiszar and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems.* Academic Press: New York, and Akademiai Kiado: Budapest, 1981.

[8] T. Berger and R. Yeung, "Multiterminal Source Encoding with One Encoder Breakdown Consideration," in preparation.

HIGH-ORDER MARKOV ARITHMETIC CODING FOR DATA COMPRESSION

Y. BAR-NESS and SEOKRIM CHOI
The Center for Communication and Signal Processing
Department of Electrical Engineering
New Jersey Institute of Technology
Newark, New Jersey 07102

ABSTRACT

Generally it is difficult to compress data, from statistically unknown sources, with significant compression ratio. To achieve this, the compression scheme should have an ability to gather context information of the source, adaptively. For better compression, it is desirable to separate the context gathering scheme from coding itself. Arithmetic code is very flexible and can accept any kind of context gathering scheme which can generate probabilities. One context gathering algorithm which generates Markov type probability adaptively is studied and implemented in software using arithmetic code. This scheme performs well and results in compression ratio better than any existing algorithms such as the Lempel-Ziv algorithm.

I. INTRODUCTION

Most computer work involves storage of large amount of data and its transmission through band limited channels. Generally alphanumerical data such as documents, papers, report have a lot of redundancy. To use transmission channel more efficiently, or to save data in smaller storage space, reduction of data redundancy is needed. This process is called Data Compression. The implementation of such a coding scheme was first described by Shannon [1]. An improved method was suggested by Huffman [2] for construction of minimum redundancy codes.

Lempel and Ziv proposed a universal coding algorithm [3], [4] which is good for any stationary ergodic sources. Later, Welsh [5] proposed an improved version of L-Z algorithm, a string of data is partitioned into its relevant parsed segments of increasing length. L-Z and LZW algorithms achieve asymptotically optimum compression adaptively. Its data gathering algorithm is very convenient, but has some drawbacks. Most of its dictionary entries are never used and it is difficult to change its data gathering process even when the data structure is known.

Data compression system is based on three basic components [6] : The model structure for context, the statistical estimation of events and the encoder (coding method). Model structures and statistics estimation are very

---

THIS WORK WAS SUPPORTED BY A GRANT FROM THE NEW JERSEY COMMISSION ON SCIENCE AND TECHNOLOGY AND A GRANT FROM AT&T INFORMATION SYSTEMS.

important, because they completely determine the maximum available compression. Actually, Huffman and Shannon's coding schemes are merely encoders. Lempel and Ziv algorithm has all three basic components mentioned above, and it is difficult to separate them. Sometimes for better compression, separate modeling and encoding of data is needed. For this purpose, the coding scheme must be very flexible. Huffman code is not flexible, because traditional Huffman codes require the design of a different code word set for different statistics.

P. Elias suggested a coding scheme [7] [8] [9] which is flexible and gives optimum compression. In Elias coding scheme the number of source alphabets and its statistics can be changed character by character. So, good source models can be used with this source coding scheme.

Of particular interest is a universal data compression scheme which can encode strings without apriori or given source probabilities. That is the source model must have capability of gathering context information 'on the fly'. Because of the properties of the languages, a context gathering algorithm which gives Markov type conditional probability is preferable. Since at any time the Markov type probability can be zero, that is it has never occurred before, a strategy is needed to change the order of Markov to find non-zero probability.

In this paper a tree structure which fits every order upto m Markov probability is made and some flags which represent change of order are used and tested. Jone's version of arithmetic code [10] is used to implement compression system with it's source model. In the next section, basic concepts of arithmetic code are introduced. In section III, context gathering algorithm is proposed and discussed, and in section IV the results of the implemented algorithm are presented.

II. Arithmetic Code

Arithmetic code is a coding technique which encodes a data (source) string by mapping it to a point on the number line between 0 and 1, and creating a code string which represents the fractional value which corresponds to that point. This coding scheme originated in Shannon's work [1]. P. Elias noticed that Shannon's idea could be improved so that the symbols need not be sorted. Elias's coding scheme is described briefly by Abramson [7], while Jelinek [8] [9] gave a detailed analysis of Elias scheme.

Elias coding scheme has a serious problem. As the message increases in length, the arithmetic involved requires higher precision. Rissanen [11] introduced an arithmetic coding procedure that avoided the need for increasing precision. His code was of the last in first out (LIFO) type. Later together with Langdon [12] he introduced an arithmetic code of first in first out (FIFO) type. The idea brought forth in this work can be looked upon as generalization of the concept of arithmetic coding. In implementing arithmetic coding in software, one has to consider all truncation and approximation that may occur during calculation and storage. Such truncation can cause fatal and propagating errors, if it occurs in unexpected location. C.B. Jones [10] proposed an integer calculation technique to avoid unexpected truncation. His coding technique is well defined and gives almost optimum compression ratio.

One of the arithmetic code's attractive features is that it needs no alphabet extension to achieve near optimum compression even for binary alphabet. This is because arithmetic code can have non-integer code length. Jones' code despite of using integer approximation, gives optimum compression ratio if it has enough precision. This is due to the inclusion of rounding-off process during calculations of code. Jones's Arithmetic code uses frequency instead of probability. By applying a scale factor, probability can be effectively converted to a frequency rate. For sources where symbol probabilities have been inferred from empirical symbol frequencies, the values can be based directly on the actual frequencies themselves.

III. Source Modeling

Arithmetic Coding system is very flexible in the sense that the probability table, the size of source alphabet and the size of code alphabet can be changed on the fly. This property enables us to encode m-th order Markov source when the probability table supplied for each character is based on the conditional probabilities of the source symbols in the state determined by the m preceding symbols. That is when a character from the source is accepted by the encoder, it is encoded by the current frequency table (probability). Immediately following that, the frequency table is updated to include the currently encoded character. Obviously in building up frequency table 'on the fly', there is a problem of initialization of probability table. If the source string had infinite length, then initialization would have little effect on compression ratio. But for the source string whose length is less than 40,000 characters, correct initialization is crucial. So a strategy to initialize the table is needed. If the table is initialized with some numbers which are not zero, the table size will become very big, and it will have unacceptable transient characteristics. For example, assume 40000- character source (8 bits per character) is being encoded and 2nd order Markov is used as a source model. To encode this source with arithmetic code, 256 * 256 * 256 conditional probabilities are initialized with non-zero numbers. While encoding the source, each probability will be increased by some amount when the corresponding symbol is met. But because of the table size, even after 40000 characters the probability table will be still in the transient state.

If the table is initialized with 0, its transient characteristics will be better, since non-zero probability character will have higher relative frequency and hence can be encoded more efficiently. In this case, however, some flags will be needed to signify all non-existing (zero probability characters. In no way arithmetic coding system can encode zero probability.

*Context Tree

Because of initialization problem and table size, tree structure is needed for the probability table.

.Constructing The Tree

Let S be a string and a_n be n-th character in the string $S = a_1 a_2 a_3 \ldots\ldots a_n \ldots$, where a_i belongs to a set of k alphabets. Let $c_{n,m} = a_{n-1} a_{n-2} a_{n-3} \ldots a_{n-m}$ be the past sequence, where $c_{n,m}$ is the past m characters of a_n in reverse order. Let T(n) be the context tree after n characters have been processed. For an m-th Markov encoding i.e. if the past sequence c has m characters, the depth of the tree is m+1. That is, it contains m+1 levels, our character set has k symbols.

1. We start with T(0). The root of T(0) has k counts (0,0,...,0). If the past sequence c_n has m characters, the depth of tree will be m+1, that is, there will be m+1 level.

2. Let T(n-1) be the current tree. After the next symbol a_n is received, the next tree T(n) is generated as follow:

- Start at the root and take the branch indicated by each of the successive symbols in the past sequence $c_{n,m}$.
- For each node visited increase the a_n component count of the node by one.
-Repeat this m times.

As an illustration, consider the string "abcabbd". And let the maximum length of $c_{n,m}$ be 5. (m = 5). The alphabet set contains 4 characters (a,b,c,d).

1. First the first level is initialized to one and all others to zero. We show only the level which has at least one non-zero entry

```
T(0)   a b c d
       | | | |
      (1,1,1,1)
```

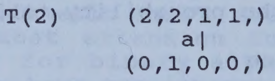
2. S = "a"
   $c_{1,5}$ = ""

   The first character is 'a'. Since no previous character exists, only the first entry in the first level is changed to 2.

   T(1) (2,1,1,1,)

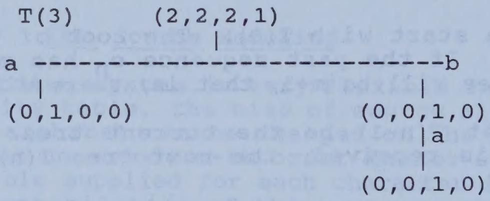
3. s = "ab"
   $c_{2,5}$ = "a"

   The letter 'b' is now processed. The second entry of the first level is increased by one. A second level node is generated with 1 at the second entry that corresponds to 'b'. On the path from first to the second level we mark the letter 'a' which represents the first letter of $c_{2,5}$.


T(2)
(2,2,1,1,)
a
(0,1,0,0,)

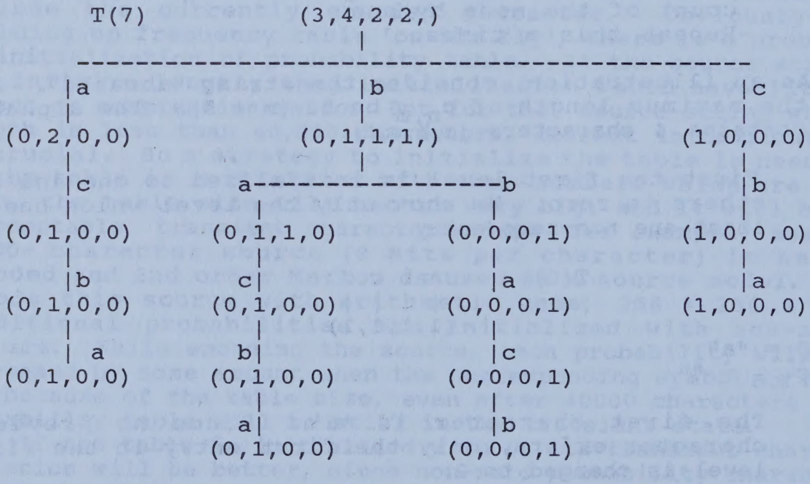
4. S = "abc"
$c_{2,5}$ = "ba"

The letter 'c' is now encoded. The third entry which corresponds to this letter is increased by one. Since the first letter in $c_{2,5}$ is 'b' then a new path has to be generated toward the second level with 'b' marked on it and the path is continued toward the third level with 'a' marked on it (The second letter in $c_{3,5}$).


T(3)
(2,2,2,1)
a
b
(0,1,0,0)
(0,0,1,0)
a
(0,0,1,0)

5. S = "abcabbd"
$c_{7,5}$ = "bbacb"

We continue in this manner to finally obtain T(7).


T(7)
(3,4,2,2,)
a
b
c
(0,2,0,0)
(0,1,1,1,)
(1,0,0,0)
c
a
b
b
(0,1,0,0)
(0,1,1,0)
(0,0,0,1)
(1,0,0,0)
b
c
a
a
(0,1,0,0)
(0,1,0,0)
(0,0,0,1)
(1,0,0,0)
a
b
c
(0,1,0,0)
(0,1,0,0)
(0,0,0,1)
a
b
(0,1,0,0)
(0,0,0,1)

<Fig. 1>

.Reading The Context Tree

In above example we made a tree. Now let's find a way to read this tree. Each node has 4 counts corresponding to 4 symbols of the source character set. The first count in each node represents the number of occurrences of symbol 'a' and second count represents 'b' and so forth. It has 6 levels because m = 5. Each branch line represents one of the characters in sequence $c_{n,m}$ according to its level. The first level of this tree has the probabilities of memoryless source. And the second level has the probabilities of the 1st Markov source, etc. Probabilities of character 'c',P(c) can be found in level 1 of (fig.1). That is P(c) = 2/11. From the second level entries we can read the conditional probability of a character whose condition is a previous character. For example, at the second entries of these nodes we have 2 corresponding to path marked 'a' 1 corresponding to path marked 'b' and 0 corresponding to path marked 'c'. Thus $P(b|a) = 2/3$, $P(b|b) = 1/3$ and $P(b|c) = 0$. And in level 3, we can find P(c/ab) = 1/2. Also by looking at the different entries of a node we can find the conditional probabilities of different characters which have the same condition. That is $P(c|b) = P(b|b) = P(d|b) = 1/3$ can be found from the entries of the second node in the second level.

*Flags

Because initial value of each node is (0,0,0,...,0) except root (first level), and different order of Markov are used according to its availability, flags are needed to communicate changes in Markov order to the decoder and to inform it of a new character which has never occurred before. There are several ways of using flags. Each way has its advantages and disadvantages. To encode flags with arithmetic code, some probalilities must be assigned to these flags. If a flag is added to a node, because it occupies some probabilities, the effective probabilities of the other symbols will be reduced resulting in reduction of the compression ratio. Several flags were tested and compared. The following one was found to give the best results.

In the context tree, each node in highest levels has a room for flag. The encoder starts searching at highest level. When it meets probability zero, it sends a /flag/, and changes the probability table to another table which has probabilities only for levels. Then it goes down the tree and searches for the non-zero probability, calculates how many levels the decoder has to move to meet nonzero probability and sends that number using the probability table for levels.

In fact the /flag/ conveys two kinds of informations. First it informs decoder which level it should use to decode the next character, and second by using the probability content of higher level nodes, it communicates the fact that some of the characters are not the ones encoded. These particular characters can be removed from the probability table of the node and hence results in more efficient encoding. For example, from (fig. 1) at node (0,1,1,0), P(c/ab) = 1/2. However, using the information on level conveyed by the flag, we know that 'b' cannot be the encoded character. Thus removing 'b' leaves (0,0,1,0) as the effective table with P(c/ab) = 1.

IV. Results and Comparison

The above-mentioned algorithms were implemented in software and tested. Arithmetic code with 4-th order source model gave the best compression. It is compared with LZW and Huffman coding scheme.

-Compression Ratio

There are a few definitions for compression ratio. In this paper compression ratio is defined as follows:

Let U: Number of bytes in un-compressed (original) file
C: Number of bytes in compressed file

Then Compression ratio is

$$\text{Compression Ratio} = (U - C) / U * 100$$

With this definition compression ratio represents the reduction ratio of redundancy. For example 70% compression ratio means that after compression 70% of original file is removed and the size of remaining compressed file is 30% of that of original file.

-Test files

Several files are used to test implemented algorithms. Because compression ratio is dependent on file's properties, we mention below a brief spec of files.

All files have the length between 16000 and 42000 characters.
ch1 - ch6 : Chapters of Electrical engineering textbook
fb1, fb2 : Technical papers
rcm : C-language manual with example programs. This file has two different text (one is English text and the other is C-program), generally it gives poor compression ratio.
sh.1 : Unformatted English text file
sh.1p : Formatted English text file (UNIX nroff output file)
compress.c: Well documented C-program file
compress : Executable file

<Table 1> shows the complete results of arithmetic coding with Markov type source model. It also shows compression ratio of LZW and Huffman coding for comparison purpose. For text files, 4-th order Markov has the best result. This might have something to do with the average length of English words.

<Fig. 2> directly compares compression ratio with different files using arithmetic code with 4th-order source and LZW. Arithmetic coding is superior by more than 10%.

<Fig. 3> shows compression ratio of arithmetic code with different order source models. The curve termed 'Without

Flag' is the maximum compression we can obtain with these models. It is a sort of upper bound for compression. The curve termed 'With Flag' is the actual compression we obtained. The difference between these two is the loss due to 'Flag'.

<Fig. 4> shows the distribution of levels in which encoding actually occurred. Maximum level (maximum length of c) is 3. And most of encoding occurs in the last level.

REFERENCES

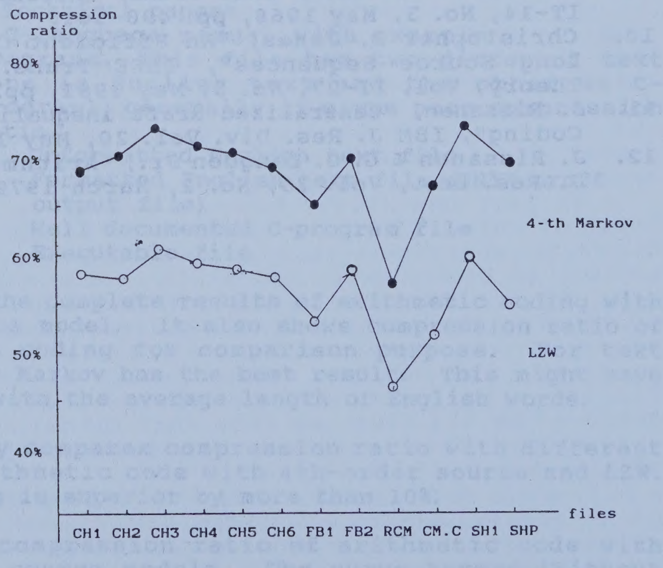
1. C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, Vol. 27, 1948, pp. 379-423, 623-656.
2. D. A. Huffman, "A Method of Constructing Minimum Redundancy Codes", Proc. IRE, Vol. 40, Sept. 1952, pp 1098-1101.
3. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Trans. Information Theory, Vol. IT-23, No. 3, May 1977, pp. 337-343.
4. J. Ziv and A. Lempel,"Compression of Individual Sequences via Variable-Rate Coding", IEEE Trans. Information Theory, Vol. IT-24, No. 5, Sept. 1978, pp. 5306.
5. T. A. Welch, "A Technique for High Performance Data Compression", IEEE Computer Journal, June 1984, pp. 8-19.
6. Glen G. Langdon Jr., "An Introduction to Arithmetic Coding", IBM J. Res. Dev., Vol. 28, No. 2, March 1984.
7. N. Abramson, Information Theory and Coding, New York: McGraw-Hill, 1969.
8. F. Jelinek, Probabilistic Information Theory, McGraw-Hill Book Company, 1968, pp. 476-489.
9. F. Jelinek, "Butter Overflow in Variable Length Coding of Fixed Rate Source", IEEE Trans. Information Theory, Vol. IT-14, No. 3, May 1968, pp. 490-501.
10. Christopher B. Jones, "An Efficient Coding System for Long Source Sequences", IEEE Trans. on Information Theory, Vol. IT-27, No. 3, May 1981, pp. 280-291.
11. J. Rissanen, "Generalized Kraft inequality and Arithmetic Coding", IBM J. Res. Div. Vol. 20, May 1976, pp. 198-203.
12. J. Rissanen & G. G. Langdon Jr., "Arithmetic Coding", IBM J. Res. Dev., Vol. 23, No. 2, March 1979, pp. 149-162.

Compression ratio

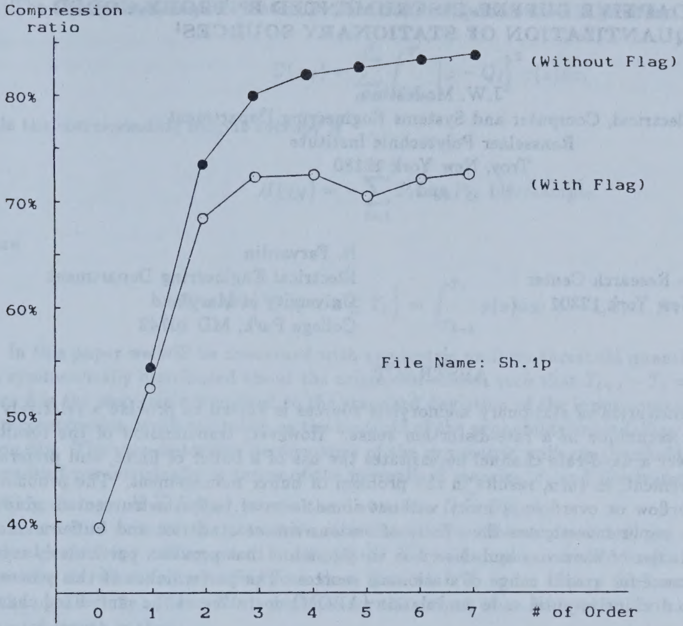
(02/27/87)

file name (Bytes)	Huff.	LZW	Arithmetic code /(# of order) 0	1	2	3	4	5	6	7
ch1 (16681)	46.7	58.0	46.7	59.1	66.5	68.9	68.9	68.7	68.5	68.4
ch2 (25615)	44.0	57.4	43.9	57.6	66.1	69.7	70.1	70.0	69.7	69.5
ch3 (15988)	47.2	60.5	47.4	62.0	69.9	72.4	72.6	72.5	72.3	72.1
ch4 (30312)	45.7	59.1	45.6	58.5	67.1	70.8	71.1	70.8	70.5	70.2
ch5 (38233)	45.3	58.3	45.3	57.4	66.0	70.1	70.5	70.2	69.9	69.7
ch6 (24476)	45.2	57.2	45.1	57.8	65.3	68.7	69.0	68.7	68.4	68.2
fb1 (14779)	44.8	52.9	44.3	55.1	62.1	65.2	65.5	65.3	65.1	64.9
fb2 (27108)	45.8	57.6	45.7	57.5	65.7	69.3	69.7	69.5	69.2	69.0
rc-read.me (16168)	38.9	46.1	38.7	49.8	55.0	57.3	57.2	57.1	57.0	57.0
compress.c (39614)	34.3	51.6	34.4	51.3	62.1	66.2	67.0	66.9	66.7	66.6
sh.1p (42181)	39.4	59.1	39.4	52.7	68.4	72.3	72.4	71.8	72.4	72.2
sh.1 (31459)	37.7	54.6	37.8	54.5	64.7	69.0	69.3	69.0	68.7	68.4
compress (34736)	31.3	49.1	32.0	50.6	59.8	62.3	63.2	63.3	63.3	63.4

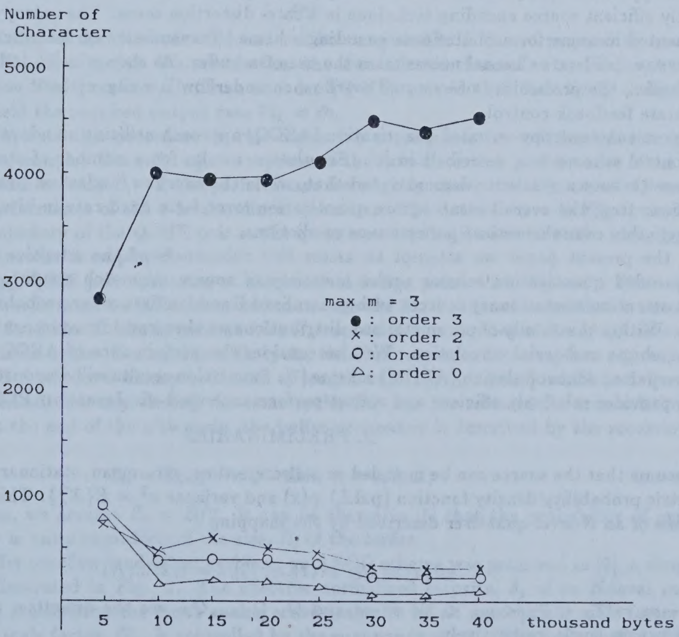
(Table 1)



(Fig.2) Comparison of Arithmetic Coding with LZW



(Fig.3) Change of compression ratio



(Fig.4) Order Distribution for every 5000 characters

ROBUST ADAPTIVE BUFFER-INSTRUMENTED ENTROPY-CODED QUANTIZATION OF STATIONARY SOURCES[1]

J.W. Modestino
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180
and

D.D. Harrison
G.E. Corporate Research Center
Schenectady, New York 12301

N. Farvardin
Electrical Engineering Department
University of Maryland
College Park, MD 20742

ABSTRACT

Entropy-coded quantization of stationary memoryless sources is known to provide a relatively efficient source encoding technique in a rate-distortion sense. However, transmission of the resulting variable-length codes over a fixed-rate channel necessitates the use of a buffer of finite, and preferably small, size. This requirement, in turn, results in the problem of buffer management. The probability of eventual buffer underflow or overflow is unity without some form of buffer-instrumented adaptive feedback control. This paper investigates the effects of various source statistics and buffer management algorithms for a buffer of fixed size and describes an algorithm that provides particularly robust rate-distortion performance for a wide range of stationary sources. The performance of this scheme is compared to conventional adaptive pulse code modulation (APCM) operating at the same fixed channel transmission rate.

I. INTRODUCTION

Entropy-coded quantization of stationary memoryless sources is well-known [1- 4] to provide a relatively efficient source encoding technique in a rate- distortion sense. The entropy coding is generally implemented as some form of Huffman encoding scheme. Transmission of the resulting variable-length codes over a fixed-rate channel necessitates the use of a buffer. As shown in [5], independent of the size of the buffer, the probability of eventual overflow or underflow is unity without some form of adaptive buffer-state feedback control.

A particular entropy-encoded quantization (AECQ) approach utilizing an adaptive buffer-state feedback control scheme was described in [6]. Simulation results for a number of stationary memoryless sources with known statistics demonstrated that, while the overflow/underflow problem can be practically eliminated, the overall mean-square quantization error for a fixed rate in bits/sample is increased only negligibly over theoretical performance predictions.

In the present paper we attempt to assess this robustness of the adaptive buffer-instrumented entropy-coded quantization scheme under a variety of source mismatch conditions. Specifically, we restrict attention to stationary sources with generalized Gaussian first-order probability density function (p.d.f.). Within this family of parameterized distributions we then consider source mismatch with respect to scale, shape and serial correlation. We also consider the performance of AECQ relative to Jayant's adaptive pulse code modulation (APCM) scheme [7]. Simulation results will demonstrate that the AECQ scheme provides relatively efficient and robust performance vis-à-vis Jayant's APCM.

II. PRELIMINARIES

Assume that the source can be modeled as a discrete-time, zero-mean, stationary process $\{X_n\}$ with symmetric probability density function (p.d.f.) p(x) and variance $\sigma^2 = E\{X_n^2\}$. This source is quantized by means of an N-level quantizer described by the mapping

$$q_N(x) = Q_\ell;\ x\epsilon(T_{\ell-1}, T_\ell], \ell = 1, 2, ..., N, \tag{1}$$

where $-\infty = T_0 < T_1 < .... < T_N = \infty$ and $Q_1, Q_2, ..., Q_N$ are the quantizer threshold levels and reconstruction levels, respectively.

---

[1] This work was supported in part by ONR under Contract N00014-75-C-0281.

The average mean-square distortion as a result of quantization is given by

$$D(q_N) = \sum_{\ell=1}^{N} \int_{T_{\ell-1}}^{T_\ell} \left[x - Q_\ell\right]^2 p(x)dx, \tag{2}$$

while the corresponding output entropy is

$$H(q_N) = -\sum_{\ell=1}^{N} P_\ell log_2 P_\ell; \text{ bits/sample}, \tag{3}$$

where

$$P_\ell = Pr\left\{T_{\ell-1} < x \leq T_\ell\right\} = \int_{T_{\ell-1}}^{T_\ell} p(x)dx; \ \ell = 1, 2, ..., N. \tag{4}$$

In this paper we will be concerned with symmetric uniform-threshold quantizers with inner thresholds symmetrically distributed about the origin and chosen such that $T_{\ell+1} - T_\ell = \delta\sigma$, $\ell = 1, 2, ..., N-2$, where δ is the step size normalized to the standard deviation of the input source process. Furthermore, the reconstruction levels are taken as the centroid of the probability mass falling within the ℓ'th quantization bin. The rate-distortion performance of the symmetric uniform-threshold quantizer can then be determined parametrically in terms of the normalized stepsize, δ, and illustrated through a plot of the resulting entropy, $H \triangleq H(q_N)$, versus the distortion, $D \triangleq D(q_N)$.

In order to achieve the attractive rate-distortion performance of uniform quantizers, some form of noiseless entropy coding must be employed. We will employ variable length Huffman codes [9, 10] exclusively in what follows. Suppose then that an L'th order binary Huffman code with codeword lengths v_ℓ, $\ell = 1, 2, ..., M = N^L$ is designed for the output of an N- level uniform quantizer. The average codeword length is then

$$\overline{m}_L = (1/L) \sum_{\ell=1}^{M} v_\ell p_\ell, \tag{5}$$

where p_ℓ is the probability of the ℓ'th codeword and can be computed from the corresponding quantizer output probabilities, P_ℓ, given by (4). We assume that the transmission channel at our disposal accepts $\hat{m}$ bits/source sample. The quantizer/Huffman encoder (Q-HE) pair is then iteratively designed, as described in [6], to yield the required output rate $\overline{m}_L = \hat{m}$.

In Fig. 1 we illustrate this procedure in rate-distortion coordinates for a Gaussian source with $N = 9$ and $\hat{m} = 2$ bits/sample. The solid square represents the rate-distortion performance of an Q-HE pair designed in this fashion with $L = 2$; the performance lies slightly above the solid curve representing the rate-distortion performance of the 9-level uniform quantizer. The dotted curve in Fig. 1 represents the rate-distortion trajectory of the Q- HE pair as the quantizer stepsize is varied but the same nominal 2'nd order Huffman encoder is employed.

Consider this fixed Q-HE pair operating at the nominal design point with $\overline{m}_L = \hat{m}$. The variable-length Huffman codewords must be buffered to accommodate the fixed channel transmission rate. Suppose that a buffer of fixed size B bits is used and let Z_n represent the number of occupied cells at the end of the n'th operation cycle. During the n'th cycle, the Huffman encoder accepts a sequence of L quantizer outputs and produces a binary codeword of length $m_n \epsilon \{v_1, v_2, ..., v_M\}$ which is delivered as input to the buffer. Simultaneously, during the same cycle, $L\hat{m}$ bits are released for transmission over the channel. Thus, at the end of the n'th cycle, the buffer occupancy is described by the recursion

$$Z_n = Z_{n-1} + m_n - L\hat{m}; \ n = 1, 2, ..., \tag{6}$$

where, for definiteness, we assume $Z_0 = B/2$. It can be shown in [5] that the probability of eventual overflow or underflow is unity regardless of the size, B, of the buffer.

To avoid this buffer overflow/underflow problem, an AECQ scheme was proposed in [6]; a simplified version of which is illustrated in Fig. 2. The effective normalized stepsize, δ, of an N-level uniform quantizer is varied by multiplying the input analog source samples by an appropriate scale factor prior to quantization. The scale factor, G_n, is controlled by the occupancy state of the finite buffer of length B bits. The quantizer output is then entropy encoded using a fixed L-level Huffman encoding scheme to provide $\overline{m}_L = \hat{m}$ at the nominal operating point. By adjusting the uniform quantizer stepsize we hope to generate a rate-distortion trajectory, similar to that illustrated in Fig. 1, which will be effective in controlling the buffer overflow/underflow problem.

While a number of feedback strategies are possible for controlling the stepsize, we will assume that the functional relationship between the quantizer stepsize, Δ_n, at the n'th operation cycle and the buffer occupancy state, Z_{n-1}, at the end of the $(n-1)$'st cycle is described as

$$\Delta_n = \Delta_0 \cdot f(Z_{n-1} - B/2);\ n = 1, 2, ..., \tag{7}$$

where Δ_0 is the stepsize at the nominal design point. The corresponding feedback gain factor in Fig. 2 then satisfies $G_n^{-1} = f(Z_{n-1} - B/2)$. In what follows, the function $f(\cdot)$ is assumed to possess the saturation characteristic

$$f(x) = \begin{cases} \frac{1}{1-\gamma/2} & ; x > K, \\ \frac{1}{1-\gamma x/2K} & ; 0 \leq x \leq K \\ 1 + \gamma x/2K & ; -K \leq x \leq 0 \\ 1 - \gamma/2 & ; x < -K, \end{cases} \tag{8}$$

where $0 \leq \gamma \leq 2$ and $K \leq B/2$ is a specified integer. Note that, because of the discrete nature of the argument of $f(\cdot)$ in (8), the stepsize, Δ_n, can assume one of $2K+1$ possible values determined by Δ_0, γ and K. Furthermore, in the absence of channel errors, the receiver can track the occupancy state of the transmitter buffer and hence no side information needs to be transmitted to implement this reconstruction scheme.

Given a fixed buffer size, B, choice of parameters γ and K for the AECQ scheme is similar to that discussed in some detail in [6]. In the experimental results to be described, we restrict attention to the choices $K = B/4$ and $1 \leq \gamma \leq 2$ which have been found to provide a good compromise between the probability of overflow/underflow and the resulting long-term average distortion.

While the original motivation underlying the AECQ is an attempt to manage the buffer overflow/underflow problem, it's of some interest to investigate the relative robustness of this scheme under deviations from nominal design conditions compared to other competing approaches. One such approach is backward adaptive PCM (APCM), as illustrated in Fig. 3. Here the stepsize, Δ_n, of an N-level uniform quantizer is modified on the basis of past quantizer outputs to strike a compromise between overload and granular quantization noise effects. We will restrict attention to the case of $N = 2^{\hat{m}}$ with a fixed channel transmission rate of $\hat{m}$ bits/sample.

The sequence $\{Y_n\}$ in Fig. 3 represents the indices of the quantizer output levels at the corresponding iteration cycle which can then be encoded using fixed-length codewords of length $m_n = \hat{m}$ bits/sample. The particular APCM scheme introduced by Jayant [7] adjusts the step size of an N-level quantizer on the basis of the most recent output according to

$$\Delta_n = M(|Y_{n-1} - (N+1)/2| + 1/2) \cdot \Delta_{n-1};\ n = 1, 2, ..., \tag{9}$$

where $M(k) = M_k, k = 1, 2, ..., N/2$, represents the so-called step-size multiplier function. Considerations relating to the choice of multipliers, $M_k, k = 1, 2, ..., N/2$, are found in [7, 11]. Optimization is very complicated and depends upon the source p.d.f. as well as steady-state and transient performance criterion. In what follows we will assume these multipliers are chosen to provide near-optimum steady-state performance for a nominal zero-mean i.i.d. Gaussian input. Under these conditions, the multipliers shown in Table I have been determined to provide near optimum performance. These will be used exclusively in what follows.

Another parameter which affects APCM performance is the allowed step size dynamic range. For good performance under mismatched conditions, the normalized step size, δ_n, must be allowed to seek average values different from the nominal value, δ_0. However, if the step size is unconstrained, unlikely source sequences of large or small values can bias the step size leading to excessively high average distortions. In what follows, we limit the normalized step size at any time n to $10^{-3} \leq \delta_n/\delta_0 \leq 100$.

III. SOURCE MODELING ASSUMPTIONS

In the simulation results to be presented we will restrict attention to stationary sources with first-order p.d.f. described by the generalized Gaussian distribution

$$p(x;\alpha) = \frac{\alpha\eta(\alpha,\beta)}{2\Gamma(1/\alpha)} exp\{-[\eta(\alpha,\beta)|x|^{\alpha}]\}, \tag{10a}$$

where $\eta(\alpha,\beta) \triangleq \beta^{-1}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2}$ with $\alpha > 0$ describing the exponential rate of decay, β a positive quantity representing a scale parameter and $\Gamma(\cdot)$ is the Gamma function. The variance of the associated random variable is given by $\sigma^2 = \beta^2$. For $\alpha = 2$ we have the Gaussian distribution while for $\alpha = 1$ this reduces to the Laplacian distribution.

Suppose then that the nominal source distribution, $p_0(x)$, is chosen within this class by selecting the parameter $\alpha = \alpha_0$ and variance $\sigma^2 = \sigma_0^2$. Mismatch to scale then occurs when the actual distribution, $p(x)$, differs only in the source variance; i.e., $p(x) = (\sigma_0/\sigma)p_0((\sigma_0/\sigma)x)$. Similarly, under mismatch to shape we assume that $p(x)$ has the same variance as $p_0(x)$ but possibly a different exponential decay parameter, i.e., $p(x) = p(x;\alpha)$ with α not necessarily equal to α_0. Finally, under mismatch to serial correlation we assume the AECQ is designed under an independence assumption when, in reality, the source possesses a rather explicit form of serial dependence. In particular, we assume the source output is a stationary first- order autoregressive (AR) process described by the recursion

$$X_n = \rho X_{n-1} + W_n;\ n = 1,2,..., \tag{11}$$

where $\{W_n\}$ is an i.i.d. sequence with p.d.f. $p_W(x)$, ρ is the correlation coefficient with values $0 \leq |\rho| \leq 1$ and $\{X_n\}$ possesses stationary first-order p.d.f. described by (10) for some fixed value of α. In what follows we consider only the case

$$p_W(x) = \frac{1}{\sqrt{2\pi(1-\rho^2)\sigma^2}} exp\left\{-\frac{x^2}{2(1-\rho^2)\sigma^2}\right\};\ -\infty < x < \infty. \tag{12}$$

The resulting stationary first-order AR process is often referred to as the Gauss-Markov process.

IV. Simulation Results:

An extensive simulation study has been made [12] of the sensitivity of both the AECQ and Jayant's APCM to source mismatch conditions. This has included tabulation of histograms of the buffer state occupancy for the AECQ as well as evaluation of the corresponding relative distortions for both the AECQ and APCM. The results of this study will be summarized here.

A.) Mismatch to Scale: In Fig. 4 we illustrate the effect of scale mismatch on the histogram of buffer state occupancy for the AECQ designed for $\hat{m} = 2$ bits/sample under the i.i.d. Gaussian assumption with $N = 9, L = 2, B = 64, K = 16$ and $\gamma = 1.0$. The curves are parameterized by the quantity $\varsigma = (\sigma/\sigma_0)^2$, representing the scale mismatch parameter. Observe that the scale mismatch in this case results in an increase (decrease) in the average occupancy level from the design point $Z_0 = B/2$ for $\varsigma > 1 (\varsigma < 1)$. This result is easily explained. For $\varsigma > 1 (\varsigma < 1)$, the nominal or design stepsize, Δ_0, achieving $\hat{m} = 2$ bits/sample is too small (large) by the factor $\sqrt{\varsigma}$ for the actual source possessing larger (smaller) variance than assumed. Thus, the quantizer average output rate is greater (less) than the channel rate and the average buffer content increases (decreases). The average buffer occupancy stabilizes when, through the feedback function in (7), the stepsize satisfies $(\Delta/\Delta_0)^2 = \varsigma$. Only in this case will the average input and output buffer rates be equal. As a result, the nominal or average operating point of the buffer must shift to the right (left) to allow, according to (7), an increase (decrease) in the average stepsize.

It should be noted that, from (8) variations in (Δ/Δ_0) are bounded by $(1-\gamma/2) \leq (\Delta/\Delta_0) \leq (1-\gamma/2)^{-1}$ so that for $\gamma = 1.0$ we would expect effective operation of the AECQ only for $0.5 \leq \varsigma \leq 4.0$. For ς outside this range the buffer-state feedback simply saturates and overflow or underflow must eventually occur. In the experiments, overflow/underflow was avoided only for $0.4 \leq \varsigma \leq 2.5$. This dynamic range is somewhat less than predicted but is easily justified by the fact that the argument was based upon average buffer occupancy whereas the histogram clearly indicates some dispersion about these average values.

One obvious method of improving the ability of the AECQ to cope with scale mismatch is to increase the value of γ. For example, with $\gamma = 2.0$ we would expect to be able to handle sources with mismatch parameter as large as $\varsigma = \infty$ and as small as $\varsigma = 0$. In Fig. 5 we indicate results for $\gamma = 2.0$ but otherwise parameters are the same as in Fig. 4. Here we observe that buffer overflow/underflow has been avoided for $0.1 \leq \varsigma \leq 16.0$. Performance for larger input variances is not achieved because, for the particular Huffman code used here, the longest codeword can cause $Z_n > B$ (overflow), even when $Z_{n-1} < K$. This problem can be avoided, however, by increasing B for fixed K. Similar comments apply to the behavior for $\gamma < 1$.

Perhaps of more interest is the distortion performance as a function of scale mismatch. In Fig. 6 we illustrate this for a Gaussian source operating at $\hat{m} = 2$ bits/sample and employing an $N = 9$ level quantizer. We illustrate the AECQ performance for both $\gamma = 1.0$ and $\gamma = 2.0$ over the range of scale mismatch for which buffer overflow/underflow was avoided. We have also included the corresponding performance of the APCM encoder described earlier as well as the rate-distortion function bound, computed as in [8], and the Lloyd-Max quantizer performance. Clearly, for the chosen set of AECQ parameters, the APCM encoder copes well with a wider range of scale mismatch. However, in the stable operating region, the AECQ does provide significant improvements in distortion relative to the APCM encoder.

The performance of an AECQ will depend heavily on the choice of feedback function as well as the available buffer length. We have illustrated results for a particular $f(x)$ but even in this case it should be clear that a larger buffer operating with a corresponding increased value of K would result in a much wider range of mismatched operation with improved average distortion. To demonstrate this we have also included in Fig. 6 the average distortion as a function of scale mismatch for $B = 128$ and 256, both with $\gamma = 2.0$. Observe the significant performance improvements over the corresponding $B = 64$ case.

Other choices of feedback control strategy, as described in [12, 13], have also resulted in significant improvements in the ability of the AECQ to cope with scale mismatch. From these results it can be concluded that a properly designed AECQ can provide improved performance relative to APCM in the face of scale mismatch.

B.) Mismatch to Shape: In Fig. 7 we illustrate typical effects of shape mismatch on the normalized distortion for the AECQ designed for $\hat{m} = 2$ bits/sample with $N = 9$. More specifically, the nominal design assumes an i.i.d. Gaussian ($\alpha_0 = 2$) source and results are provided as the characteristic exponent, α, of the generalized Gaussian source distribution varies about this nominal choice. Again, as in Fig. 6, we illustrate performance for the AECQ with both $\gamma = 1.0$ and $\gamma = 2.0$, each over their corresponding ranges of operation, as well as the performance of the APCM encoder. For reference, we also include the performance of the Lloyd-Max quantizer, matched to each value of α, and the corresponding rate-distortion performance. As can be observed from Fig. 7, for $B = 64$ the AECQ, with its larger number of quantizer levels and entropy coding, outperforms the non-uniform Lloyd-Max quantizer for all but the most narrow- tailed densities (e.g., $\alpha = 5.0$). It is considerably more efficient than the APCM encoder and is far less sensitive to shape mismatch. Furthermore, observe how the performance can be improved even further by going to larger B and/or N as illustrated for $B = 128, N = 25$ and $B = 256, N = 75$ in Fig. 7. In both these cases we have employed only a first-order ($L = 1$) Huffman code since higher-order codes resulted in such long codewords than much larger B would be required to avoid overflow/underflow. The results for $B = 256, N = 75$, in particular, track the $R(D)$ bound quite closely.

C.) Mismatch to Serial Correlation: An illustration of the effects of serial correlation on the normalized distortion occupancy is provided in Fig. 8. Here, an i.i.d. Gaussian source is assumed and an $N = 9$ level quantizer is employed operating at $\hat{m} = 2$ bits/sample. The nominal design choice then corresponds to a first-order Gaussian AR source, as defined by (11), with $\rho = \rho_0 \triangleq 0$. Again we have illustrated performance of the AECQ for both $\gamma = 1.0$ and $\gamma = 2.0$ and have included the performance of the APCM encoder. Observe that, for the range of ρ where overflow/underflow was avoided, the distortion with the AECQ was much lower than with APCM. We can see from these results that the AECQ is an attractive alternative to APCM under serial correlation mismatch.

V. SUMMARY AND CONCLUSIONS

We have provided a detailed study of the performance of AECQ in the presence of source mismatch conditions and have compared the relative performance to its closest competitor, APCM. Our results indicate that the AECQ can provide considerably improved performance and more robust operation than APCM. The relative sensitivity of the AECQ to scale mismatch is directly related to the rather simple memoryless feedback control strategy we have employed. This might be remedied by alternative choices of memoryless feedback control function, $f(\cdot)$, or by introducing memory. The former is much easier to implement and/or analyze than the latter.

The results of this work leads to further questions, whose answers are the focus on ongoing work [12]. For example, given an unknown scale mismatch, it may be that there is a feedback function which is in some sense optimum for a given buffer length, B. Also, it should be possible to completely avoid buffer overflow and underflow, as in [3], by effectively installing reflecting boundaries at the ends of the buffer. This leads to some form of bit stuffing when near underflow, and to suboptimum quantization when near overflow. This could be combined with the feedback mechanism to provide improved performance. Finally, since the sequence $\{Z_n\}$ is a Markov chain, it should be possible to calculate the buffer state histograms and distortions directly, instead of indirectly through simulation of randomly generated source sequences. In the meantime, the results reported here demonstrate the potential of the AECQ in providing efficacious and robust performance for a wide variety of stationary sources.

REFERENCES

1. J.R. Goblick and J.L. Holsinger, "Analog Source Digitization: A Comparison of Theory and Practice", IEEE Trans. Inform. Theory, Vol. IT-13, pp. 323-326, Apr. 1967.
2. H. Gish and J.N. Pierce, "Asymptotically Efficient Quantization", IEEE Trans. Inform. Theory, Vol. IT-14, pp. 676-683, Sept. 1968.
3. T. Berger, "Minimum Entropy Quantizers and Permutation Codes", IEEE Trans. Inform. Theory, Vol. IT-28, pp. 149-157, Mar. 1982.
4. N. Farvardin and J.W. Modestino, "Optimum Quantizer Performance for a Class of Non-Guassian Memoryless Sources", IEEE Trans. Inform. Theory, Vol. IT-30, pp. 485-497, May 1984.
5. N. Farvardin and J.W. Modestino, "On the Overflow and Underflow Problems in Buffer-Instrumented Variable-Length Coding of Fixed-Rate Memoryless Sources", IEEE Trans. Inform. Theory, Vol. IT-32, pp, 516-532, Nov. 1986.
6. N. Farvardin and J.W. Modestino, "Adaptive Buffer-Instrumented Entropy-Coded Quantizer Performance for Memoryless Sources", IEEE Trans. Inform. Theory, Vol. IT-32, pp. 9-22, Jan. 1986.
7. N.S. Jayant, "Adaptive Quantization with a One Word Memory", Bell Syst. Tech. J., pp. 1119-1144, Sept. 1973.
8. R.E. Blahut, "Computation of Channel Capacity and Rate-Distortion Functions", IEEE Trans. Inform. Theory, Vol. IT-18, pp. 460-473, July 1972.
9. R.G. Gallager, Information Theory and Reliable Communication, Wiley, New York, 1968.
10. D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proc. IEEE, Vol. 40, pp. 1098-1101, Sept. 1952.
11. N.S. Jayant and P. Noll, Digital Coding of Waveforms, Prentice-Hall, Englewood Cliffs, NJ, 1984.
12. D.D. Harrison,"Adaptive Entropy-Encoded Data Compression Techniques", Ph.D. Thesis, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, in preparation.
13. J.W. Modestino, "Comments on Adaptive Buffer-Instrumented Entropy- Coded Quantization Scheme Under Mismatch Conditions", TM 86-9, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, Sept. 1986.

M_k \ N	4	8	16
M_1	0.80	0.90	0.90
M_2	1.60	0.90	0.90
M_3	-	1.25	0.90
M_4		1.75	0.90
M_5		-	1.20
M_6			1.60
M_7			2.00
M_8			2.40

Table I
Step-size Multipliers for
Memoryless Gaussian Source.

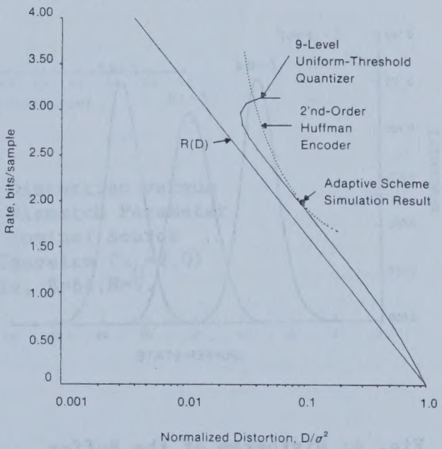


Fig. 1: Rate-Distortion Performance of the Symmetric Uniform Threshol Quantizer and Corresponding Huffman Encoder for a Memoryless Source with Gaussian Distribution: $\hat{m}=2$ bits/sample,N=9,L=2.

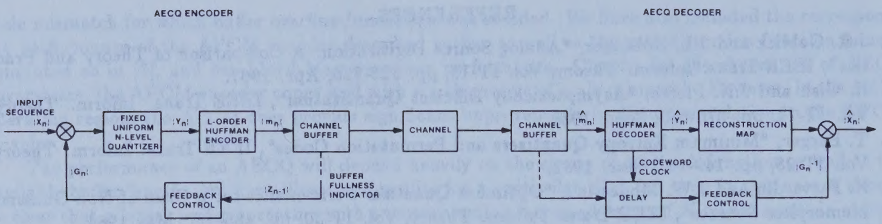

AECQ ENCODER
AECQ DECODER
INPUT SEQUENCE
$\{X_n\}$
FIXED UNIFORM N-LEVEL QUANTIZER
$\{Y_n\}$
L-ORDER HUFFMAN ENCODER
$\{m_n\}$
CHANNEL BUFFER
CHANNEL
CHANNEL BUFFER
$\{\hat{m}_n\}$
HUFFMAN DECODER
$\{\hat{Y}_n\}$
RECONSTRUCTION MAP
$\{\hat{X}_n\}$
$\{G_n\}$
FEEDBACK CONTROL
$\{Z_{n-1}\}$
BUFFER FULLNESS INDICATOR
CODEWORD CLOCK
DELAY
FEEDBACK CONTROL
$\{G_n^{-1}\}$

Fig. 2: Block Diagram of the Simplified Version of the Adaptive Entropy-Coded Quantizer (AECQ) Encoder/Decoder Pair.

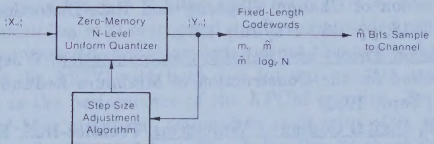
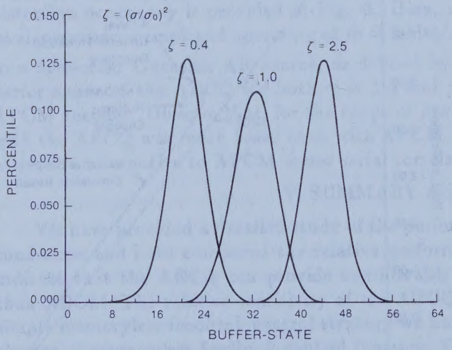

$\{X_n\}$
Zero-Memory N-Level Uniform Quantizer
$\{Y_n\}$
Fixed-Length Codewords
m_n $\hat{m}$
$\hat{m}$ $\log_2 N$
$\hat{m}$ Bits Sample to Channel
Step Size Adjustment Algorithm

Fig. 3: Backward Adaptive PCM (APCM) Encoder.



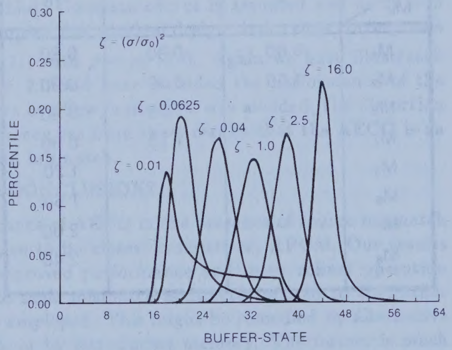


Fig. 4: Histogram of the Buffer State in the AECQ Under Scale Mismarch. Nominal Source Gaussian $(\alpha_0 = 2.0)$ with $\hat{m}=2$ bits/sample, N=9 $\gamma=1.0$.

Fig. 5: Histogram of the Buffer State in the AECQ Under Scale Mismatch. Nominal Source Gaussian $(\alpha_0 = 2.0)$ with $\hat{m}=2$ bits/sample, N=9 $\gamma=2.0$.

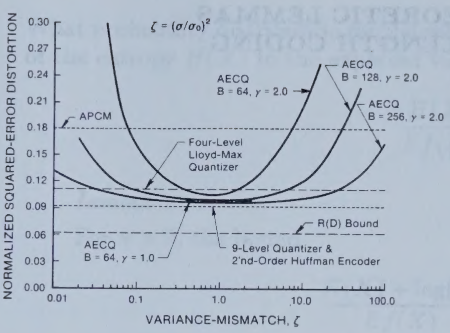


Fig. 6: Normalized Distortion versus Scale Mismatch Parameter for AECQ and APCM. Nominal Source Gaussian(α_0=2.0) with $\hat{m}$=2 bits/sample. AECQ Employs N=9 Level Quantization.

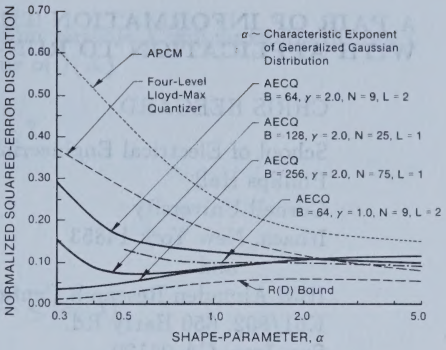


Fig. 7: Normalized Distortion versus Shape Mismatch Parameter for AECQ and APCM. Nominal Source Gaussian(α_0=2.0) with $\hat{m}$=2 bits/sample.

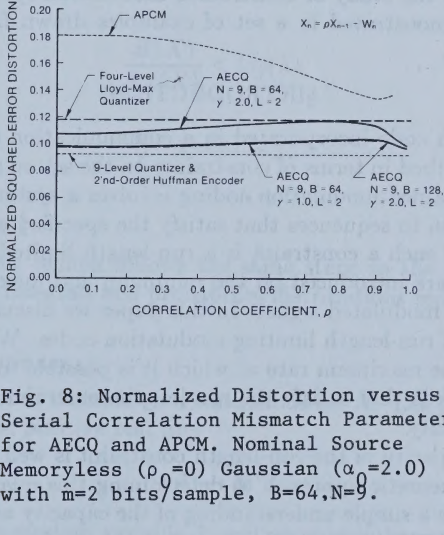


Fig. 8: Normalized Distortion versus Serial Correlation Mismatch Parameter for AECQ and APCM. Nominal Source Memoryless (ρ_0=0) Gaussian (α_0=2.0) with $\hat{m}$=2 bits/sample, B=64,N=9.

A PAIR OF INFORMATION THEORETIC LEMMAS WITH APPLICATION TO RUN-LENGTH CODING

CHRIS HEEGARD

School of Electrical Engineering
Phillips Hall
Cornell University
Ithaca, New York 14853
and
IBM Almaden Research Center
K61/802, 650 Harry Rd.
San Jose, CA 95120

ABSTRACT

This paper presents a pair of information theoretic lemmas. The lemmas describe the maximum value that the ratio of the entropy of a random variable X to the expected value of a positive function of the random variable $Ef(X)$ can take. The optimum distribution for X is also explicitly provided. These lemmas are useful tools in the study of constrained modulation systems. The utility of the lemmas is demonstrated in a set of examples drawn from the problem of run-length coding.

§INTRODUCTION

A modulation code incorporated in a communication or recording system can often be described in terms of constraints on the set on allowed signals. The problem of constrained modulation coding involves a systematic mapping from unconstrained data to sequences that satisfy the specified constraint. An interesting example of such a constraint is a run-length limited code [1, 4]. In this case, constraints are introduced on the minimum and maximum time between transitions in the modulated signal. In this paper we discuss information theoretic properties of run-length limiting modulation codes. We consider the problem concerning the maximum rate at which it is possible to map unconstrained data onto codewords that satisfy the run-length constraint; this maximum rate is called the capacity.

While the capacity of the run-length constraint is well known [2], we study an information theoretic approach to determining this quantity as in [4]. This technique presents a simple understanding of the capacity and shines interesting new light on this problem. A key component to this analysis involves the solution to a basic question in information theory. This question comes in two flavors: discrete and continuous.

§TWO LEMMAS

Question 1 (discrete):

Let X be a random variable on a discrete set $\mathcal{X}$. Let $f(x)$ be a function from the set $\mathcal{X}$ to the positive real line

$$f : \mathcal{X} \rightarrow \Re^+.$$

What probability distribution for the random variable X will maximize the ratio of the entropy $H(X)$ to the expected value of $f(X)$

$$\frac{H(X)}{Ef(X)}?$$

Lemma 1

For $\gamma > 0$, the bound

$$\frac{H(X)+\log(\gamma)}{Ef(X)} \leq log(\lambda)$$

holds with equality if and only if $Pr(X = x) = \gamma\lambda^{-f(x)}$ where

$$\sum_{x\in\mathcal{X}} \gamma\lambda^{-f(x)} = 1.$$

Note that this last equation uniquely determines $\lambda > 0$.

Corollary 1

$$\frac{H(X)}{Ef(X)} \leq log(\lambda)$$

where

$$\sum_{x\in\mathcal{X}} \lambda^{-f(x)} = 1.$$

Proof of Lemma 1

The proof of this lemma follows the same steps as the proof of lemma 2 with sums replacing integrals and probability distributions replacing probability density functions.

Question 2 (continuous):

Let X be a random variable on a continuous set $\mathcal{X}$. Let $f(x)$ be a function from the set $\mathcal{X}$ to the positive real line

$$f : \mathcal{X} \to \Re^{+}.$$

What density for the random variable X will maximize the ratio of the differential entropy $h(X)$ to the expected value of $f(X)$

$$\frac{h(X)}{Ef(X)}?$$

Lemma 2

For $\gamma > 0$, the bound

$$\frac{h(X)+\log(\gamma)}{Ef(X)} \leq log(\lambda)$$

holds with equality if and only if the density of X, $p(x) = \gamma\lambda^{-f(x)}(a.e.)$, where

$$\int_{\mathcal{X}} \gamma\lambda^{-f(x)}\, dx = 1.$$

Again note that this last equation uniquely determines $\lambda > 0$.

Corollary 2

$$\frac{h(X)}{Ef(X)} \le log(\lambda)$$

where

$$\int_{\mathcal{X}} \lambda^{-f(x)}\, dx = 1.$$

Proof of Lemma 2

The proof consists of two simple steps.

First, let X^* be the random variable with the density $q(x) = \gamma\lambda^{-f(x)}$. Then by definition

$$\begin{aligned} h(x) &\equiv -\int_{\mathcal{X}} q(x)\log(q(x))\, dx \\ &= \int_{\mathcal{X}} q(x)(f(x)\log(\lambda) - \log(\gamma))\, dx \\ &= Ef(X)\log(\lambda) - \log(\gamma). \end{aligned} \tag{1}$$

Now, the *discrimination* (or *Kullback-Liebler number*) for probability densities is defined by [3]

$$D(p\|q) \equiv \int_{\mathcal{X}} p(x)\log\left(\frac{p(x)}{q(x)}\right) dx.$$

It is easy to show (and well known) that this quantity is non-negative and equal to zero if and only if $p(x) = q(x)(a.e.)$. (The proof follows

$$\begin{aligned} &\int_{\mathcal{X}} p(x)\log\left(\frac{p(x)}{q(x)}\right) dx \\ &\ge \log(e)\int_{\mathcal{X}} p(x)\left(1 - \frac{q(x)}{p(x)}\right) dx \\ &= \log(e)\int_{\mathcal{X}} p(x) - q(x)\, dx = \log(e)(1-1) = 0 \end{aligned}$$

where the inequality follows from $\log(y) \ge \log(e)(1 - \frac{1}{y})$ with equality if and only if $y = 1$.) Thus if X has density $p(x)$ and X^* has density $q(x) = \gamma\lambda^{-f(x)}$, it follows that

$$0 \le D(p\|q) = -h(X) + \int_{\mathcal{X}} p(x)(\log\left(\gamma\lambda^{-f(x)}\right) - \log(\gamma))\, dx$$

$$= -h(X) + Ef(X)\log(\lambda) - \log(\gamma)$$

or

$$h(X) + \log(\gamma) \leq Ef(X)\log(\lambda).$$

Since $f(x)$ is a positive function, both $Ef(X)$ and $Ef(X^*)$ are positive and

$$\frac{h(X) + \log(\gamma)}{Ef(X)} \leq \log(\lambda) = \frac{h(X^*) + \log(\gamma)}{Ef(X^*)}.$$

The last equality follows from equation (1). Note that equality holds only when $p(x) = q(x) = \gamma\lambda^{-f(x)}(a.e.)$.

This completes the proof of lemma 2.

§APPLICATIONS OF THE LEMMAS

Run-Length Codes: Discrete, M-level Modulation

Communication and storage systems often transmit or store information via a *modulation signal*, w(t). A synchronous, pulse amplitude modulation (PAM) signal takes the form of a weighted sum of shifted versions of a *modulation pulse* p(t)

$$w(t) = \sum_{j=1}^{n} w_j p(t - j\Delta).$$

In this signal, the *modulation (or amplitude) sequence* w_j belongs to a *modulation alphabet*, $\mathcal{W}$, of M letters (i.e., $|\mathcal{W}| =$ M). The interval Δ is called the *symbol period* of the system.

A *modulation encoder* is a device that maps unconstrained binary information onto the modulation sequence of the signal $w(t)$. Often, the sequence is required to satisfy certain constraints (e.g., power, spectrum or run-length constraints). These constraints limit the number of available modulation sequences and thus the rate at which the modulation encoder can operate. If we let $A_n \subset \mathcal{W}^n$ denote the set of modulation sequences of length n that satisfy the required constraints, then the *noiseless capacity* is defined as the limit

$$C \equiv \lim_{n\to\infty} \frac{1}{n} \log|A_n|.$$

One approach to determining the capacity for a given constraint is to apply combinatorial techniques to count the number of valid sequences (e.g., for M-ary (d, k) codes, see [5]). A different approach is explored here.

The capacity problem can be placed in an information theory setting by associating a probability model with the encoder. For example, if random binary data is used at the input of the encoder, then the output sequences are a random process. The *rate* of the encoder is then equal to the entropy of the output process

$$H(W) \equiv \lim_{n\to\infty} \frac{1}{n} H(W_1, W_2, ... W_n).$$

The capacity question is then equivalent to determining the maximum entropy that the output process can assume. In determining the capacity in this manner, the maximum entropy (or maxentropic) distribution is obtained. This distribution is helpful in understanding the constrained modulation problem because it describes the composition of the typical output sequences when an efficient (i.e., near capacity) encoder is used. Note that by the concavity of the entropy function, the maxentropic distribution corresponds to a modulation sequence that is a stationary random process. Therefore, in the following discussion assume that W_j is stationary.

The usefulness of Lemma 1 in determining the value of the capacity is demonstrated for the class of run-length constraints. A modulation code (i.e., the set sequences produced by an encoder) that satisfies a (d, k) *run-length constraint* $(d < k)$ is often described in terms of the binary *transition sequence* x_j of the modulation sequence w_j. The sequence is defined by $x_0 \equiv 1$ and for $j > 0$

$$x_j = \begin{cases} 0, & \text{if } w_j = w_{j-1}; \\ 1, & w_j \neq w_{j-1}. \end{cases}$$

The sequence satisfies the (d, k) constraint if whenever there is a transition, $x_j = 1$, then there are at least d non-transitions

$$x_{j+1} = x_{j+2} = \ldots = x_{j+d} = 0$$

and at most k non-transitions

$$x_l = 1, \quad \text{for some } l = j+d+1, j+d+2, \ldots, j+k+1.$$

This constraint can be visualized by looking at figure 1.

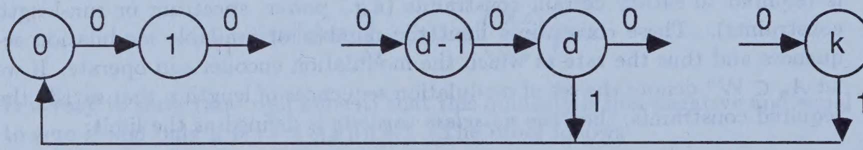


Figure 1: (d, k) Transition sequences as labels of a graph.

The capacity of the $(d < k)$ constraint [5], $C = \log(\lambda)$, where λ is the positive root of the equation

$$\lambda^{k+1} = (M-1)(1 + \lambda + \ldots + \lambda^{k-d}) \tag{2a}$$

or for $k = \infty$

$$\lambda^{d+1} = (M-1) + \lambda^d. \tag{2b}$$

This result is readily obtained from Lemma 1.

Note that the entropy of the modulation sequence

$$H(W) = H(W, X) = H(X) + H(W|X) \tag{3}$$

(where $H(W|X)$ is the conditional entropy) since the transition sequence is a function of the modulation sequence. Thus the capacity question is one of maximizing the sum in equation (3). The second term satisfies the inequality

$$H(W|X) \leq p_x(1)\log(M-1) \tag{4}$$

where $p_x(1)$ is the frequency of transitions (i.e., the probability that $X_j = 1$). The inequality follows from

$$H(W_1, W_2, ..., W_n|X_1, X_2, ..., X_n) = \sum_{j=1}^{n} H(W_j|X_1, X_2, ..., X_n, W_1, W_2, ..., W_{j-1})$$

$$\leq \sum_{j=1}^{n} H(W_j|X_j, W_{j-1}) = \sum_{j=1}^{n} Pr(X_j = 1)H(W_j|X_j = 1, W_{j-1}).$$

Note by stationarity $Pr(X_j = 1) = p_x(1)$ and when $X_j = 1$, $H(W_j|X_j = 1, W_{j-1}) \leq \log(M-1)$ since $W_j \neq W_{j-1}$. Equality is achieved when the conditional distribution of W_j given $X_j = 1$ is uniformly distributed over the set $\mathcal{W} - \{w\}$ when $W_{j-1} = w$.

The maxentropic distribution is best described in terms of the run-length values themselves. This is the approach of Zehavi and Wolf [4]. Define the transition times $t_0 = 0$ and for $j > 0$

$$t_j = \min\{j > t_{j-1} \mid x_j = 1\}.$$

Then the run-length sequence is defined by the differences

$$T_j \equiv t_j - t_{j-1}.$$

For a (d, k) run-length constraint, the runlengths must satisfy the bounds

$$d + 1 \leq T_j \leq k + 1.$$

In terms of the run-length process, it follows that

$$H(X) = \frac{H(T)}{E(T)}$$

and

$$p_x(1) = \frac{1}{E(T)}.$$

It follows from (3) that

$$H(W) = \frac{H(T) + \log(M-1)}{E(T)} \tag{5}$$

when equation (4) is satisfied with equality. This last equation, (5), is maximized by run-lengths that are independent and identically distributed (i.i.d.) with a distribution chosen to maximize

$$\frac{H(T_1) + \log(M-1)}{E(T_1)}.$$

The maximizing distribution is provided by Lemma 1. In this case, $\gamma = M - 1$, the capacity $C = \log(\lambda)$ and the optimum distribution

$$Pr(T_1 = j) = (M-1)\lambda^{-j}.$$

The equation for λ is

$$\sum_{j=d+1}^{k+1} (M-1)\lambda^{-j} = 1$$

which can be seen to be equivalent to equation (2).

Run-Length Codes: Discrete, Other Examples

Other examples of discrete run-length constraints are easily handled by Lemma 1. For these examples, we take the number of levels, $M = 2$ (generalization to other values of M are straightforward); we need only apply Corollary 1.

Asymmetric (d, k) Codes

Consider a run-length code with a separate constraint on the even and odd run-lengths. Specifically, a (d_0, k_0, d_1, k_1) code satisfies

$$T_{2j} \in \{d_0+1, d_0+2, ..., k_0+1\}, T_{2j+1} \in \{d_1+1, d_1+2, ..., k_1+1\}$$

Then the capacity is determined by maximizing

$$\frac{H(T_1)+H(T_2)}{E(T_1)+E(T_2)} \leq \log(\lambda) = C$$

where

$$(\sum_{j=d_0+1}^{k_0+1} \lambda^{-j})(\sum_{j=d_1+1}^{k_1+1} \lambda^{-j}) = 1.$$

Note that the optimum distribution involves independent runlengths that are alternately drawn according to two distributions (one for even, one for odd).

Sum (d, k, s) Codes

Consider a run-length code with a constraint on the sum of the even and odd run-lengths. Specifically, a (d, k, s) code satisfies

$$T_{2j}, T_{2j+1} \in \{d+1, d+2, ..., k+1\}, T_{2j} + T_{2j+1} \leq s.$$

Then the capacity is determined by maximizing

$$\frac{H(T_1, T_2)}{E(T_1)+E(T_2)} \leq \log(\lambda) = C$$

where

$$\sum_{j=d+1}^{k+1} \sum_{j=d+1}^{min(k+1,s-j)} \lambda^{-(j+l)} = 1$$

(note in Corollary 1, $X = (T_0, T_1)$ and $f(X) = T_0 + T_1$). The optimum distribution involves i.i.d. pairs of runlengths; the pairs themselves are dependent.

Run-Length Codes: Continuous

Run-length codes maintain a certain minimum and maximum time difference between transitions in the modulation signal w(t). For example, in terms of time intervals, for a fixed symbol period Δ and (d, k) code, the run-length intervals

$$T_j \in \{(d+1)\Delta, (d+2)\Delta, ..., (k+1)\Delta\}$$

(note the change of notation; the run-lengths are now multiples of the symbol period, Δ, and not integers). In this way, the runlength are bounded between a minimum and maximum value

$$T_{min} \leq T_j \leq T_{max}$$

where $T_{min} = (d+1)\Delta$ and $T_{max} = (k+1)\Delta$. These bounds can be maintained by using a smaller symbol period, Δ, and by adjusting d and k appropriately. Specifically, if $L \geq 1$, then

$$\Delta' = \frac{\Delta}{L}, \quad d' = (d+1)L - 1, \quad k' = (k+1)L - 1$$

generate modulation signals with the same values of T_{min} and T_{max}.

In the limit, as $L \to \infty$, the run-lengths become continuous random variables. In this case, it is interesting to study the behavior of the ratio of the differential entropy to the expected length of the run-lengths. Corollary 2 describes the distribution of $T_{min} \leq T^* \leq T_{max}$ that maximizes

$$\frac{h(T^*)}{E(T^*)} = \log(\lambda).$$

In this case the density of T^* is λ^{-t} where

$$\int_{T_{min}}^{T_{max}} \lambda^{-t}\, dt = 1.$$

Note that three cases are of interest

$$T_{max} - T_{max} \begin{cases} > 1, & \lambda < 1, h(T) > 0; \\ = 1, & \lambda = 1, h(T) = 0; \\ < 1, & \lambda > 1, h(T) < 0; \end{cases}$$

(see figure 2).

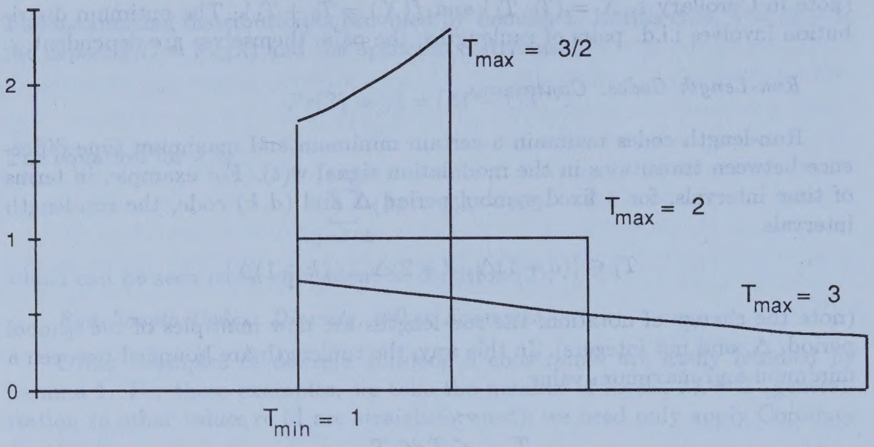


Figure 2: Run-length distribution for various T_{max}.

Acknowledgments

The author would like to acknowledge Razmik Karabed of IBM/ARC for pointing out that the discrete case (Lemma 1) follows as a special case of a lemma in Csiszár and Körner's book [6, pp. 56].

§REFERENCES

[1] Ayis Gallopoulos, Chris Heegard and Paul Siegel, "The Power Spectrum of Run-Length-Limited Codes," submitted to *IEEE Transactions on Communications.*

[2] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for Sliding Block Codes," *IEEE Transactions on Information Theory*, vol. IT-29, pp. 5-22, 1983.

[3] Richard Blahut, "Principles and Practice of Information Theory," Addison-Wesley, 1987.

[4] E. Zehavi and Jack K. Wolf, "On Run-Length Codes," to appear *IEEE Transactions on Information Theory.*

[5] Glenn S.Dixon, Catherine A. French and Jack Keil Wolf, "Results Involving (D, K) Constrained M-ary Codes," preprint.

[6] Imre Csiszár and János Körner, "Information Theory," Academic Press, 1981.

Support

This work was supported in part by NSF grant ECS-8352220, Control Data Corporation and AT&T.

A Comparison of the Wolf–Ziv and Alphabet–Constrained Approaches for Vector Quantization of Noisy Speech

JERRY D. GIBSON, THOMAS R. FISCHER, and BONEUNG KOO
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843

ABSTRACT

The minimum mean–square error encoding of a noisy source is considered within the context of alphabet–constrained data compression. Just as in the block coding formulation, the optimum source coder consists of an optimum estimator followed by optimum source coding of the resulting estimates. However, unlike the block approach, the alphabet–constrained viewpoint admits estimators based upon the past history of source observations outside of the current block of interest. In fact, if delayed encoding is allowed, the estimator is an optimum smoother. Extensive experiments are reported for vector quantization of noisy speech using the Linde, Buzo, Gray training mode vector quantizer. The results indicate that the alphabet–constrained estimator, which is a Kalman filter, is superior to the block estimator, and, in particular, that adaptivity is critical for good performance over a variety of speech sources.

I. INTRODUCTION

The design of optimum source codes for a noisy source has a lengthy history [1–5, 16], with the major result being that the minimum mean square error (MSE) encoder for a block of noisy source samples decomposes into an optimum block estimator followed by an optimum source coder operating on the estimator output [2]. The results in [2] are based upon a block coding formulation where each block is treated independently, which was common in the source coding literature in 1970 and is still one of the standard approaches today. In this paper we show that the block coding formulation places undue restrictions upon the problem solution, and that by employing the alphabet–constrained approach [5], the class of admissible source coders is broadened. In particular, the optimum coder for noisy sources under the alphabet–constrained formulation still decomposes into an optimum estimator followed by an optimum source coder, but now the optimum estimator can take advantage of dependence between blocks.

The block coder performance approaches the distortion rate function of the noisy source as the blocklength $N \to \infty$ [2], and for a fixed, finite N, the block coder is optimum for a one–shot use of the channel [4]. Unfortunately, for most data compression applications, we wish to compress a very long sequence of source samples using a relatively short blocklength ($N \leq 10$), which is not the problem addressed by the block approach. On the other hand, the alphabet–constrained coder structure, which exploits any block–to–block dependence, is ideally suited to this problem. In fact, the alphabet–constrained coder always performs at least as well as the block coder. Here we consider the important problem of vector quantization of noisy speech and contrast the performance of three block estimators and five alphabet–constrained (Kalman) estimators.

II. PROBLEM FORMULATION

A. Block Coding Formulation

Following [2], the communication system is diagrammed as in Figure 1. A discrete–time source produces a sequence $X(i)$, which is modified by a source mapping to produce the sequence $S(i)$. The encoder maps $S(i)$ into a sequence $Y(i)$ for transmission over a communications channel. The received sequence, $\hat{Y}(i)$, is mapped by the decoder to $\hat{S}(i)$, which in turn is modified by a receiver mapping to $\hat{X}(i)$, which is delivered to the final destination. Only the sequence $S(i)$ is available to the encoder, and the basic design problem is to select an encoder/decoder pair to minimize the mean-square error between $X(i)$ and $\hat{X}(i)$.

Wolf and Ziv [2] used a block coding problem formulation. Specifically, let N consecutive source elements be collected to form the vector $\mathbf{X} = (X(1), \ldots, X(N))^T$, with vectors $\mathbf{S}$, $\mathbf{Y}$, $\hat{\mathbf{Y}}$, $\hat{\mathbf{S}}$, and $\hat{\mathbf{X}}$ defined similarly. Each vector is an element in its respective sample space, $\mathbf{X} \in \Omega$, $\mathbf{S} \in \Gamma$, $\mathbf{Y} \in \Upsilon$, $\hat{\mathbf{Y}} \in \hat{\Upsilon}$, $\hat{\mathbf{S}} \in \hat{\Gamma}$, and $\hat{\mathbf{X}} \in \hat{\Omega}$. The encoder, f, maps Γ into Υ, while the decoder, g, maps $\hat{\Upsilon}$ into $\hat{\Gamma}$. Finally, define the distortion measure as

$$D_N = E\left\{ \tfrac{1}{N}\|\mathbf{X} - \hat{\mathbf{X}}\|^2 \right\} = E\left\{ \tfrac{1}{N}\textstyle\sum_{i=1}^{N}(X_i - \hat{X}_i)^2 \right\}.$$

Wolf and Ziv then showed that the minimum distortion, D_N^*, may be expressed as

$$D_N^* = E\left\{ \frac{1}{N}\|\mathbf{X} - \mathbf{U}\|^2 \right\} + \inf_{f,g} E\left\{ \frac{1}{N}\|\hat{\mathbf{X}} - \mathbf{U}\|^2 \right\} \tag{1}$$

where $\mathbf{U} = E(\mathbf{X}|\mathbf{S})$. We note two features of the solution in Eq. (1). First, as noted in [2], if the receiver mapping is the identity, then as $N \to \infty$, D_N^* approaches the distortion-rate function for the noisy source. Second, for a fixed and finite value of N, D_N^* is the optimum performance for a one-shot use of the channel [4]. Typically in data compression applications, however, one wishes to compress a large number of source samples, but using a fairly small vector dimension (block size N). For example, in the clustering approach to vector quantizer (VQ) design [6], because of the computational burden of encoding, the vector dimension rarely exceeds six. The block coding formulation does not treat the encoding of consecutive (noisy) source vectors, and it is this problem that we now address.

B. The Alphabet-Constrained Formulation [5]

Let the discrete-time sequences $X(i)$, $S(i)$, etc. in Figure 1 be assembled as sequences of N-dimensional vectors, $\mathbf{X}(n)$, $\mathbf{S}(n)$, etc. Corresponding to time step n, the encoder selects a transmission vector, $\mathbf{Y}(n)$, based on its information set, $\Psi_e(n)$. Similarly, the decoder produces an output $\hat{\mathbf{S}}(n)$, based on its information set, $\Psi_d(n)$. These information sets are described explicitly as

$$\Psi_e(n) = \left\{ \mathbf{S}(n+k), \mathbf{S}(n+k-1), \ldots, \mathbf{S}(n), \mathbf{S}(n-1), \ldots ; \mathbf{Y}(n-1), \mathbf{Y}(n-2), \ldots \right\} \tag{2}$$

and

$$\Psi_d(n) = \left\{ \hat{\mathbf{Y}}(n+m), \hat{\mathbf{Y}}(n+m-1), \ldots, \hat{\mathbf{Y}}(n), \ldots ; \hat{\mathbf{S}}(n-1), \hat{\mathbf{S}}(n-2), \ldots \right\} \tag{3}$$

and represent, respectively, the observations and past encoder outputs available to the encoder, and the received vectors and past decoder outputs available to the decoder. Selecting the parameter $k > 0$ allows for a k step look-ahead, or delayed encoding, while selecting $m > 0$ allows for delayed decoding. Consistent with the previous section, we seek an encoder/decoder pair to minimize the block mean-square error distortion measure, D_N.

The difference between the block coding formulation and the alphabet-constrained formulation is now apparent in the information available to the encoder and decoder. In the former case, only $\mathbf{S}(n)$ is used by the encoder to determine the transmission vector $\mathbf{Y}(n)$ and only $\hat{\mathbf{Y}}(n)$ is used by the decoder to produce $\hat{\mathbf{S}}(n)$. Obviously $\Psi_e(n)$ and $\Psi_d(n)$ are supersets of $\mathbf{S}(n)$ and $\hat{\mathbf{Y}}(n)$ and so an optimum encoder/decoder pair for the alphabet-constrained formulation must perform at least as well as for the block coding formulation.

The optimum encoder/decoder pair for the alphabet-constrained formulation achieves a distortion

$$D_N^* = E\left\{ \frac{1}{N} \|\mathbf{X} - E(\mathbf{X}|\Psi_e)\|^2 \right\} + \inf_{f,g} \left[E\left\{ \frac{1}{N} \|E\{E[\mathbf{X}|\Psi_e]|\Psi_d\} - E[\mathbf{X}|\Psi_e]\|^2 \right\} + E\left\{ \frac{1}{N} \|\hat{\mathbf{X}} - E\{E[\mathbf{X}|\Psi_e]|\Psi_d\}\|^2 \right\} \right]. \tag{4}$$

The proof follows directly from the orthogonality of the conditional–mean estimate and the estimation error.

The difference between Eq. (4) and Eq. (1) is the conditioning information in the conditional expectations, corresponding to the more general encoder and decoder information sets. The basic structure of the source coding system is thus unchanged, but the performance may be considerably different, as demonstrated in [17] and in the following section.

III. VECTOR QUANTIZATION OF NOISY SPEECH

Let the source, $X(n)$, be generated according to the autoregressive (AR) model

$$X(i) = \sum_{j=1}^{P} a_j X(i-j) + W(i) \tag{5}$$

where $W(i)$ is a zero–mean, white Gaussian process of variance σ_w^2. The source mapping is taken to be an additive white Gaussian noise sequence, $V(i)$, of variance σ_v^2, so that the noisy source samples presented to the encoder are

$$S(i) = X(i) + V(i). \tag{6}$$

As in the previous section, N consecutive samples of $X(i)$ (or $S(i)$) are blocked together as the vector $\mathbf{X}(n)$ (or $\mathbf{S}(n)$). The optimum source coding system of Eq. (4) should then form the conditional mean,

$$\mathbf{U}(n) = E[\mathbf{X}(n)|\Psi_e(n)] \tag{7}$$

and then optimally encode vector $\mathbf{U}$. Design of (locally) optimum vector quantizers for a given source, $\mathbf{U}(n)$, can be accomplished with the clustering algorithm of Linde, Buzo, and Gray [6]. For the Gaussian souce in Eq. (5) corrupted by additive white Gaussian noise as in Eq. (6), we consider the optimum block estimator given by

$$\mathbf{U}_B(n) = E[\mathbf{X}(n)|\mathbf{S}(n)] = \Lambda_X[\Lambda_X + \sigma_v^2 I]^{-1}\mathbf{S} \tag{8}$$

where $\Lambda_X = \mathrm{cov}[\mathbf{X}]$, and the optimum estimator from the alphabet–constrained formulation, which is

$$\mathbf{U}_A(n) = E[\mathbf{X}(n)|\mathbf{S}(n),\ \mathbf{S}(n-1), \ldots]. \tag{9}$$

Comparisons of the theoretical performance of these estimators for various Gauss–Markov sources are present in [17]. Here we use these estimators to attack a problem of significant current interest, namely, vector quantization of noisy speech. Since there are numerous variations and subtleties involved in applying the theory to a particular application, we separate the present development into subsections entitled the Block Estimator, the Kalman Estimator, Estimator Comparisons, and Overall Coder Performance. Besides clarity of presentation, these divisions make sense from the theoretical results in Section II which say that the optimum encoder in the presence of noise is an optimal estimator followed by optimum coding of the estimator output. Furthermore, from [17] for Gauss–Markov sources, as the rate (R) increases, the distortion is lower bounded by the estimator performance.

We present results for three objective performance measures in this section, the estimator SNR given by

$$\mathrm{SNR_E} \triangleq 10\log_{10}\frac{\langle X^2(n)\rangle}{\langle[X(n)-U(n)]^2\rangle}, \tag{10}$$

where $\langle\cdot\rangle$ denotes a time averaging over the utterance, the VQ SNR,

$$\mathrm{SNR_{VQ}} \triangleq 10\log_{10}\frac{\langle U^2(n)\rangle}{\langle[U(n)-Y(n)]^2\rangle}, \tag{11}$$

and the overall encoder SNR,

$$\mathrm{SNR_O} \triangleq 10\log_{10}\frac{\langle X^2(n)\rangle}{\langle[X(n)-Y(n)]^2\rangle}. \tag{12}$$

The five sentences used in this work are described in Appendix A. We define the input SNR as

$$\mathrm{SNR_{in}} \triangleq 10\log_{10}\frac{\langle X^2(n)\rangle}{\sigma_v^2}, \tag{13}$$

and we present results here only for $\mathrm{SNR_{in}} = 5$ dB in order to limit the number of cases considered.

A. Block Estimator

When considering the implementation of the block estimator in Eq. (8), one is immediately faced with the problem of how to choose Λ_X, the covariance of the uncorrupted speech, when only the noisy samples are available, and to a lesser extent, how to select σ_v^2, the noise variance. Addressing the simpler problem first, we could assume that we know σ_v^2 from a priori information or we could calculate an estimate of σ_v^2 from noisy measurements when the source is known to be zero (if such a situation presents itself). For the studies here on the block estimator, we assume that σ_v^2 is known.

The choice of Λ_X is somewhat more problematical, and hence, we study the performance of the block estimator for three different cases. In all cases we have assumed that the speech is zero mean and thus set Λ_X equal to an autocorrelation matrix.

Case 1: We calculate Λ_X based upon the uncorrupted speech itself, which serves as a bound on the best performance possible using the block approach.

Case 2: We compute Λ_X based upon a training set of uncorrupted speech that does not include the utterance to be processed.

Case 3: We set Λ_X equal to the average autocorrelation matrix constructed from McDonald's average speech data [8].

It is clear that as we progress from Case 1 to Case 3, we are examining a successively more realistic configuration.

Tables I–III present $\mathrm{SNR_E}$ as a function of blocklength (N) for the block estimators corresponding to Cases 1–3, respectively. Sentence 1 data is common to all three tables, and in comparing the results for Sentence 1 in Tables I–III, two things are evident. First, $\mathrm{SNR_E}$ is monotonically increasing with N in all three cases, and second, the variation in $\mathrm{SNR_E}$ across the three tables for each block length N is slight. Therefore, as far as Sentence 1 is concerned, there is little loss in performance when a covariance matrix based upon generic average speech data is used. However, if we examine the results in Tables I and III for Sentence 4, we see that while the data in Table I is monotonically increasing with N, the $\mathrm{SNR_E}$ values in Table III behave erratically as N increases and are several tenths of a dB less than the values in Table I for $N \geq 4$. Furthermore, the

$\mathrm{SNR_E}$ values for Sentences 2,3, and 5 in Table III behave much the same as those for Sentence 1 and are significantly greater than the Sentence 4 $\mathrm{SNR_E}$ results.

The reason for this behavior revolves around how well the autocorrelation terms of the average speech data [8] match the autocorrelations at various lags of the individual sentences. Table IV lists the autocorrelation terms of Sentences 1–5 and of McDonald's average speech data [8] for lags 1 through 8. Clearly, the average speech data autocorrelations follow those of Sentences 1, 2, 3, and 5 rather well, but differ noticeably from the autocorrelations of Sentence 4. Thus, this mismatch in Λ_X causes the relatively poor performance for Sentence 4 in Table III. Another point of interest is that the $\mathrm{SNR_E}$ for Sentence 4 in the matched case (Table I) is significantly less than the $\mathrm{SNR_E}$ values for the other sentences when $N \geq 2$. The cause of this performance difference seems to be that the variation in the autocorrelations over the entire utterance is relatively greater for Sentence 4 than for the other sentences, so that the fixed Λ_X over the utterance limits $\mathrm{SNR_E}$ performance. Therefore, one important conclusion that can be drawn from the results in this subsection is that an average covariance matrix may function quite well for many utterances, but for a few sentences, the $\mathrm{SNR_E}$ may be significantly degraded.

B. Kalman Filter

In order to implement the optimal estimator in Eq. (9), we begin with the assumed AR source model

$$X(n) = \sum_{i=1}^{P} a_i X(n-i) + gW(n) \tag{14}$$

where $W(n)$ is an independent, identically distributed, zero mean, unit variance Gaussian sequence, the coefficients $\{a_i, i = 1,2,\ldots,P\}$ are parameters to be computed or selected, and the model order $P = 4$. Now, a direct implementation of Eq. (9) would have the form

$$\begin{aligned} E\{\mathbf{X}(n)|\mathbf{S}(n),\mathbf{S}(n-1),\ldots\} = [&E\{X(nN)|S(nN),S(nN-1),\ldots;\mathbf{Y}(n-1),\ldots\},\\ &E\{X(nN-1)|S(nN),S(nN-1),\ldots;\mathbf{Y}(n-1),\ldots\},\ldots,\\ &E\{X(nN-N+1)|S(nN),S(nN-1),\ldots;\mathbf{Y}(n-1),\ldots\}]^T, \end{aligned} \tag{15}$$

but we choose instead to form an N-vector estimator from the components

$$E\{X(nN)|S(nN),S(nN-1),\ldots;\mathbf{Y}(n-1),\ldots\} \tag{16a}$$

$$E\{X(nN-1)|S(nN-1),\ S(nN-2),\ldots;\mathbf{Y}(n-1),\ldots\} \tag{16b}$$

$$\vdots$$

$$E\{X(nN-N+1)|S(nN-N+1),\ S(nN-N),\ldots;\mathbf{Y}(n-1),\ldots\}. \tag{16c}$$

Comparing Eq. (16a) with the first component of Eq. (15), Eq. (16b) with the second component of Eq. (15), and Eq. (16c) with the last component of Eq. (15), we see that while the first component of (15) and (16a) are the same, the other quantities are different. In particular, the given conditions in Eqs. (16b) and (16c) are subsets of the conditioning on the components in Eq. (15). As a result, the estimators in Eq. (16) can perform no better than those in Eq. (15). Therefore, although estimators based upon Eq. (16) are simpler and more straightforward, we are absorbing some attendant loss in performance. As we shall see, however, results based upon the estimators in Eq. (16) still establish the superiority of the alphabet-constrained approach over the standard block formulation.

Proceeding to the development of the Kalman filtering algorithms, we let $N = P$ and write a canonical state space model from Eq. (14) as

$$\mathbf{X}(n) = \mathbf{F}\mathbf{X}(n-1) + \mathbf{G}W(n) \tag{17}$$

where

$$\mathbf{X}^T(n) = [X(n-P+1)X(n-P+2)\ldots X(n)], \tag{18}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & & & & 0 \\ 0 & 0 & & & 0 & 1 \\ a_P & a_{P-1} & & \ldots & \ldots & a_1 \end{bmatrix} \tag{19}$$

and

$$\mathbf{G}^T = [0\;0\;\ldots\;0\;g]. \tag{20}$$

In conjunction with Eq. (17), we have the observation or measurement model

$$S(n) = \mathbf{H}^T\mathbf{X}(n) + V(n) \tag{21}$$

with

$$\mathbf{H}^T = [0\;0\;\ldots\;0\;1], \tag{22}$$

so that Eq. (21) corresponds to Eq. (6).

From standard Kalman filtering theory [7, 9] the state vector estimate is

$$\hat{\mathbf{X}}(n) = \mathbf{F}\hat{\mathbf{X}}(n-1) + \mathbf{K}(n)[S(n) - \mathbf{H}^T\mathbf{F}\hat{\mathbf{X}}(n-1)] \tag{23}$$

with the initial condition $\hat{\mathbf{X}}^T(0) = [0\;\ldots\;0]$ and

$$\mathbf{K}(n) = \mathbf{P}_X(n|n-1)\mathbf{H}[\sigma_v^2 + \mathbf{H}^T\mathbf{P}_X(n|n-1)\mathbf{H}]^{-1} \tag{24}$$

where

$$\mathbf{P}_X(n|n-1) = \mathbf{F}\mathbf{P}_X(n-1)\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \tag{25}$$

and

$$\mathbf{P}_X(n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{H}^T]\mathbf{P}_X(n|n-1). \tag{26}$$

With the initial condition, $\mathbf{P}_X(0) = [0]_{N\,X\,N}$, Eq. (25) is processed first, then Eq. (24) followed by Eqs. (23) and (26). The speech sample estimate at time instant n is finally obtained from

$$\hat{X}(n) = \mathbf{H}^T\hat{\mathbf{X}}(n). \tag{27}$$

In the block estimator approach, we have a dilemma as to how to select Λ_X, and for the Kalman estimator, we have a similar problem concerning how to pick $\mathbf{F}$, or more precisely, the coefficients $\{a_i,\ i = 1, 2, \ldots, P\}$, and $\mathbf{G}$. Similar to Cases 1 and 2 for the block estimator, we consider the following cases:

Case 4: We calculate one set of coefficients averaged over the entire uncorrupted utterance, which is an idealized situation.

Case 5: We compute the coefficients based upon a training set of uncorrupted speech that does not include the utterance to be processed.

In both Cases 4 and 5, we select g^2 of $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ to be a constant equal to the mean squared prediction error for the sentence being processed and the appropriate set of coefficients.

Using the Case 4 approach with a fourth order model ($P = 4$) on Sentence 1 yields $\mathrm{SNR_E} = 7.627$ dB for $\mathrm{SNR_{in}} = 5$ dB, and using the Case 5 approach on Sentence 1 with coefficients trained on Sentences 2,3, and 4 gives $\mathrm{SNR_E} = 7.460$ dB. Note that for the suboptimal Kalman filtering method applied here, the estimator does not depend on the encoder (VQ) blocklength.

Another possibility for determining the coefficients $\{a_i\}$ would be to adaptively compute them on a framed basis, where a frame is 120 samples (say), from the noisy speech measurements, and then filter this same block with the Kalman filter in Eqs.

(23)–(27). This approach of computing coefficients from noisy speech and then using these coefficients in a filter has the disadvantage of using coefficients calculated from corrupted speech, but it has the advantage of adapting the coefficients over the utterance. This concept has been previously suggested and employed in similar applications [10–12], although algorithm details differ. Here we use the standard autocorrelation method from LPC analysis to compute the coefficients from each frame of the noisy speech [13–15].

Noting that our overall goal is vector quantization of noisy speech and that if we use the training mode VQ of [6], the design process is off-line, we can attempt to improve our filtering performance by iterating through the sentence several times as suggested by Lim and Oppenheim [10]. Explicitly what we mean by iterating is that we compute coefficients from noisy speech and then use these coefficients in the Kalman filter. Rather than stopping here, however, we can compute new coefficients from the filtered speech and employ these coefficients in the Kalman algorithm to filter the speech again, and the process is repeated.

For ease of discussion, we label the preceding possibilities as:

Case 6: We compute coefficients from noisy speech on a frame-by-frame basis and use these coefficients in the Kalman filter.

Case 7: We start with Case 6 and iterate.

Case 8: As a bound on the best performance possible, we compute the coefficients on a frame-by-frame basis from uncorrupted speech and use these "noise-free" coefficients in the Kalman filter.

For simplicity, we only present results for fourth order models ($P = 4$) and $\mathrm{SNR_{in}} = 5$ dB. The $\mathrm{SNR_E}$ values for Cases 6, 7, and 8 are given in Table V.

Several interesting points are evident in Table V. Clearly, iterating improves $\mathrm{SNR_E}$, but after two iterations, $\mathrm{SNR_E}$ drops slightly. Informal listening tests confirm that the first iteration can improve subjective performance, but additional iterations, while reducing the hissing due to the additive noise, actually distort the desired speech spectrum. Also evident from Case 8 in Table V is that significant improvements in $\mathrm{SNR_E}$ are possible for the Kalman filter algorithm with a good state space model ($\mathbf{F}$ and $\mathbf{G}$).

C. Estimator Comparisons

Comparisons of the results from Sections IV.A and IV.B yield several important conclusions. First, for Sentences 1 and 5, the block estimator performs quite well compared to the Case 6 Kalman results (also Cases 4 and 5), with the block estimator producing nearly the same or larger $\mathrm{SNR_E}$ when the (blocklength) $N \geq 4$ for Sentence 1 and $N \geq 6$ for Sentence 5, even when the McDonald average covariance is used (Table III). One iteration with the Kalman filter reverses this advantage, with a clear advantage in $\mathrm{SNR_E}$ for Case 7 over Cases 1, 2, and 3. For Sentence 4, however, the Kalman estimator without iterating (Case 6) has a higher $\mathrm{SNR_E}$ than the block estimator with $N = 8$, even when the Λ_X matrix is trained on the uncorrupted sentence. This is because the autocorrelation varies significantly over the duration of Sentence 4 and the frame-adaptive Kalman estimator models this variation.

Subjectively, it is fair to state from our listening tests that differences of $\mathrm{SNR_E}$ less than 1 dB are generally not audible, so (most of) the results for Sentence 1 and Cases 1–6 sound about the same. When the difference exceeds 1 dB, however, a difference in subjective performance can be heard. The objective $\mathrm{SNR_E}$ values have the most meaning subjectively for Sentence 4. The Kalman estimators for Cases 6, 7, and 8 have a noticeable subjective advantage over the fixed parameter approaches in Cases 1–5. Thus, the frame-adaptive technique which meshes nicely with the Kalman filter is especially important.

Comparing the results for Sentences 1,2,3, and 5 with those for Sentence 4, accentuates a behavior that is something of a truism in speech processing; namely, algorithms based upon average data perform well for most speech utterances and speakers (say 85–90%), but these average designs can have a noticeable loss in performance for a smaller, but significant portion (10–15%, say) of utterances and speakers. For general use, then, speech processors (filters and coders here) must be designed to have good performance for this smaller population of speakers and utterances as well.

D. Overall Coder Performance

Due to length limitations, we present overall simulation performance results consisting only of SNR_O for Sentences 1 and 4 and Cases 3 and 6. Additionally, VQ performance for uncorrupted Sentences 1 and 4 are given. The VQs are designed using the Linde, Buzo, Gray algorithm [6], and we limit consideration to dimensions (N) 1 through 8 for rate $R = 1$ bit/dimension and dimensions 1 through 4 for rate $R = 2$ bits/dimension, since encoding complexity grows in proportion to 2^{RN}. All of this data is presented in Table VI.

For Sentence 1, there is very little difference in performance between the block estimator with an average, fixed Λ_X and the frame-adaptive Kalman estimator when $R = 1$, $N \geq 5$ and $R = 2$, $N \geq 3$. This data thus supports the use of the simpler block estimator. On the other hand, for Sentence 4 the Kalman estimator has a clear advantage over the block estimator for all R and N shown. Of course, these conclusions follow those for SNR_E and point out the importance of a speech coder being adaptive to respond to variations in speech utterances and speakers.

Subjectively, it is found that for $R = 1$, $N \leq 4$ and $R = 2$, $N \leq 2$, the VQ performance dominates the output speech quality and little performance difference is audible among the several estimators (Cases 1–6). As the VQ distortion lessens, which occurs when SNR_O approaches SNR_E, the performance afforded by the various estimators begins to dominate, with differences in SNR_O of near 1 dB being audible.

The VQ results for uncorrupted speech are included merely as a reference point. Note, however, that SNR_VQ for Sentence 4 is significantly smaller than for Sentence 1. We believe that this is a consequence of the variation in statistics across Sentence 4, and it implies that an adaptive VQ would be beneficial.

IV. CONCLUSIONS

The minimum mean-square error encoding of a noisy source has been considered in the context of alphabet-constrained data compression. The alphabet-constrained approach admits a wider and generally better performing class of estimators than does the standard block coding formulation. Three block estimators and five Kalman estimators are compared for use with training mode vector quantization of noisy speech. The SNR values and subjective listening test results indicate that the Kalman estimator is preferable to the block estimator because of its ability to adapt to variations in utterances and speakers.

REFERENCES

[1] Dobrushin and Tsybakov, *IRE Trans. Inform. Th.*, Vol. IT-8, pp. 293–304, 1962.

[2] Wolf and Ziv, *IEEE Trans. Inform. Th.*, Vol. IT-16, pp. 406–411.

[3] Berger, *Rate Distortion Theory*, Prentice-Hall, 1971.

[4] Witsenhausen, *IEEE Trans. Inform. Th.*, Vol. IT-26, pp. 518–521, 1980.

[5] Gibson and Fischer, *IEEE Trans. Inform. Th.*, Vol. IT-28, pp. 443–457, 1982.

[6] Linde, Buzo, and Gray, *IEEE Trans. Commun.*, Vol. COM-28, pp. 84–85, 1980.

[7] Gelb, *Applied Optimal Estimation*, The MIT Press, 1974.

[8] McDonald, *Bell System Tech. J.*, pp. 1123–1151, 1966.

[9] Sage and Melsa, *Estimation Theory with Applications to Communications and Control*, McGraw-Hill, 1971.

[10] Lim and Oppenheim, *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-26, pp. 197–210, June 1978.

[11] Gibson, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-10, pp. 529–536, Aug. 1980.

[12] McVay and Gibson, *IEEE Trans. Acoust., Speech, and Sig. Proc.*, Vol. ASSP-30, pp. 436–439, June 1982.

[13] Markel and Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, 1976.

[14] Makhoul, *Proc. IEEE*, Vol. 63, pp. 561–580, Apr. 1975.
[15] Rabiner and Schafer, *Digital Processing of Speech Signals*, Prentice–Hall, 1978.
[16] Ephraim and Gray, *IEEE Trans. Inform. Th.*, to appear.
[17] Fischer and Gibson, Conf. Rec., 1986 IEEE Int. Conf. on Commun., pp. 1868–1872.

APPENDIX A

SPEECH DATA

The speech data used in the simulation studies consists of the following five sentences:
Sentence 1: "The pipe began to rust while new." (female speaker)
Sentence 2: "Add the sum to the product of these three." (female speaker)
Sentence 3: "Oak is strong and also gives shade." (male speaker)
Sentence 4: "Thieves who rob friends deserve jail." (male speaker)
Sentence 5: "Cats and dogs each hate the other." (male speaker)

Each sentence was low pass filtered to 3200 Hz (3 dB), sampled 8000 times/sec., and quantized to 12 bits/sample to obtain a dynamic range of ±2048. The output low pass filter was a seven pole elliptic filter with a 2880 Hz. (3 dB) cutoff frequency.

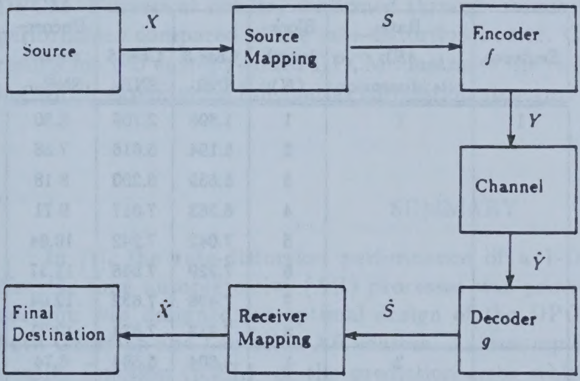


Figure 1. Communication System.

TABLE I. SNR_E VERSUS BLOCKLENGTH FOR CASE 1

	Sentence	
N	1	4
1	6.189	6.164
2	7.296	6.785
3	7.773	7.085
4	8.109	7.181
5	8.358	7.278
6	8.527	7.399
7	8.662	7.418
8	8.747	7.498

TABLE II. SNR_E VERSUS BLOCKLENGTH FOR CASE 2

	Sentence
N	1
1	5.977
2	7.169
3	7.657
4	7.984
5	8.254
6	8.434
7	8.559
8	8.630

TABLE III. SNR_E VERSUS BLOCKLENGTH FOR CASE 3

	Sentence				
N	1	2	3	4	5
1	6.189	6.179	6.173	6.164	6.141
2	7.296	7.264	7.292	6.635	7.234
3	7.768	7.811	7.796	6.888	7.631
4	8.097	8.211	8.173	6.800	7.880
5	8.326	8.422	8.422	6.831	8.035
6	8.463	8.621	8.582	6.948	8.131
7	8.558	8.679	8.675	6.876	8.195
8	8.634	8.782	8.762	6.912	8.258

TABLE IV. NORMALIZED AUTOCORRELATION TERMS FOR SENTENCES 1–5 AND MCDONALD'S AVERAGE SPEECH DATA

Sentence	Autocorrelation Terms							
	r(1)	r(2)	r(3)	r(4)	r(5)	r(6)	r(7).	r(8)
1	.8682	.5986	.2797	–.0427	–.2558	–.3657	–.3693	–.2784
2	.8639	.5417	.1671	–.1537	–.3270	–.3430	–.2548	–.1278
3	.8625	.5703	.2262	–.1022	–.3274	–.4408	–.4264	–.3233
4	.7326	.3883	.1718	–.0714	–.1029	–.0618	–.1580	–.2258
5	.8591	.6051	.3612	.0829	–.1575	–.2855	–.3369	–.3202
McDonald	.8644	.5570	.2274	–.0297	–.1939	–.2788	–.3030	–.2823

TABLE V. SNR$_E$ FOR FRAME–ADAPTIVE KALMAN FILTERING

Case	Sentence		
	1	4	5
6	8.073	7.798	8.136
7 Iteration			
1	9.237	8.701	9.330
2	9.531	8.867	9.603
3	9.551	8.827	9.579
4	9.532	8.772	9.526
5	9.504	8.721	9.483
6	9.469	8.681	9.448
7	9.436	8.656	9.418
8	9.925	9.171	9.989

TABLE VI. OVERALL SYSTEM PERFORMANCE COMPARISON

Sentence	Rate (R) bits/dimension	Block–length (N)	Case 3 SNR$_0$	Case 6 SNR$_0$	Uncorr–rupted SNR$_{VQ}$
1	1	1	1.898	2.709	2.80
		2	5.194	5.616	7.58
		3	5.539	6.200	8.18
		4	6.363	7.017	9.71
		5	7.042	7.242	10.84
		6	7.229	7.568	11.37
		7	7.498	7.632	12.04
		8	7.813	7.874	12.71
	2	1	4.604	5.594	8.74
		2	6.627	6.939	12.56
		3	7.279	7.611	14.87
		4	7.806	7.873	16.22
4	1	1	1.436	2.062	2.09
		2	3.649	4.210	5.35
		3	4.088	4.656	6.08
		4	4.846	5.447	7.17
		5	5.218	5.970	8.42
		6	5.504	6.262	9.07
		7	5.864	6.844	10.10
		8	5.846	6.853	10.68
	2	1	3.867	4.684	7.40
		2	5.477	6.255	10.70
		3	6.210	6.824	12.23
		4	6.168	6.889	13.34

RATE-DISTORTION PERFORMANCE OF 2-D DPCM FOR GAUSSIAN AUTOREGRESSIVE SOURCES[1]

J.W. MODESTINO and F. AZADEGAN
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180

ABSTRACT

An approach is described for evaluating the rate-distortion performance of 2-D entropy-coded DPCM operating on 2-D Gaussian autoregressive (AR) sources. This approach is an extension of a previously developed iterative algorithm for evaluating the rate-distortion performance of 1-D entropy-coded DPCM operating on both Gaussian and non-Gaussian AR sources. The extension to 2-D is not straightforward, however, due to a number of factors, including the absence of a strict linear ordering of the plane in 2-D. Nevertheless, by restricting attention to separable Gaussian AR sources, we are able to provide a complete rate-distortion characterization of 2-D entropy-coded DPCM. Numerical results, confirmed through simulation, indicate relatively efficient performance compared to the rate-distortion bound. Comparisons with corresponding results for 1-D entropy-coded DPCM illustrates the relative advantages of 2-D processing which exploits the full spatial correlation structure of the source.

SUMMARY

In [1], the rate-distortion performance of a 1-D DPCM scheme operating on discrete-time autoregressive (AR) processes was provided. Basically, an iterative algorithm was designed for optimal design of the DPCM encoder when operating on both Gaussian and Laplacian AR sources. An assumption was made on the probability density function (p.d.f.) of the prediction error which was then used to design the quantizer. The resulting p.d.f. of the prediction error was then recalculated and the procedure repeated. This procedure is then applied iteratively until the change in the prediction error p.d.f. from one stage to the next is less than some tolerance level. The iterative procedure then halts with the corresponding rate-distortion performance announced. Using this procedure, a fairly comprehensive description of the ultimate theoretical rate-distortion performance of 1-D DPCM was provided. In more recent work [2], the accuracy of these theoretical predictions were demonstrated by computer simulation for the Laplacian case. However, all the theoretical studies conducted on DPCM so far have been limited to the 1-D case. It is of some interest at this time to extend the approach in [1] to 2-D DPCM operating on 2-D AR sources.

The extension of the approach described in [1] to 2-D is not straightforward, however, due to the absence of a strict linear ordering of the plane in 2-D. Nevertheless,

---

[1] This work was supported in part by ONR under Contract No. N00014-15-C-0281.

by restricting attention to separable 2-D AR processes we show that it is possible to suitably extend the 1-D approach.

In this paper then we present a complete characterization of the rate- distortion performance of 2-D DPCM schemes operating on 2-D separable Gaussian AR image sources. A novel iterative algorithm is developed to overcome the difficulties associated with the absence of a linear ordering of the plane and is used to design the DPCM quantizer to minimize the mean-square distortion subject to a constraint on the first-order output entropy. We have used a projection method to approximate the distribution of the prediction error. Results on the rate-distortion performance of 2-D DPCM schemes employing uniform-threshold quantizers for 2-D separable first-order Gauss-Markov sources for a wide range of row and column correlations are presented. Also, asymptotic results for high rates are developed. These numerical results, confirmed through simulation, indicate relatively efficient performance.

References

1. N. Farvardin and J.W. Modestino, "Rate-distribution performance of DPCM scheme for autoregressive sources", IEEE Trans. Inform. Theory, Vol. IT- 31, pp. 402-418, May 1985.
2. J.D. Gibson and D.G. Jeong, "Data compression of a simple composite source", Proc. 24th Annual Allerton Conference, October 1986.

HIGHLY EFFICIENT ARITHMETIC CODING SCHEMES FOR LINE DRAWINGS BASED ON CHAIN CODE REPRESENTATIONS

JAMES GEORGE DUNHAM and CHENG-CHANG LU
Department of Electrical Engineering
Southern Methodist University
Dallas, Texas 75275

SUMMARY

Line drawings have become an important area in image data processing. For applications in contours, coastlines, weather maps and geometric shapes, it is necessary to store and transmit large amounts of line drawings and process the information by digital computer [1]. Freeman [2] proposed an 8-directional encoding scheme which represents line drawings by 3 bits per link and has been widely applied to pictorial data processing.

Efforts have been made to improve the coding efficiency. Freeman [3] proposed a chain-difference coding scheme which assigns variable-length codeword to the difference between two consecutive links. A chain-difference code represents line drawings by slightly more than two bits per link on the average. Yasuda and Kimoto [4] suggested a coding scheme which achieved a higher compression rate by restricting the line drawings to be perfect 8-connected curves. Kaneko and Okudaira [5] proposed a segment coding scheme which utilized the property that a curve with gentle curvature can be divided into long segments and represented by adjacent-direction chain codes and length indexes. Experiments have shown that this takes about 40-50 percent less space than Freeman's original chain coding scheme. Dunham and Lu [7] have proposed an efficient and easily implemented Markov chain-difference coding schemes based on Freeman's chain code representation. The scheme utilized a pattern extraction method and variable-length coding techniques to compress the chain code data.

Arithmetic coding was introduced by Rissanen in 1975 and is an effective data compression technique [8]. It maps a string of data symbols into a code string which is treated as the magnitude of a number [9] and the original data source can be recovered from the code sequence by magnitude comparisons.

We consider the chain-difference code as a Markov data source and use an arithmetic code for compression. The empirical frequency of the next state given the present state was obtained by combining the chain-difference codes of the 8 line drawings of [7]. Two conditioning trees, T_1 and T_2 with the size 8 and 29, are designed based on empirical frequency distributions and the symmetry of the chain-difference code. The T-conditioning entropy, $H(T_1)$ and $H(T_2)$, and performance bounds, $E(T_1)$ and $E(T_2)$, are calculated according to [10]. The experimental results are shown in Table I.

We can view the combined empirical frequency of the 8 line drawings as a universal model for compressing the line drawings. We seek to find the best universal model that works well for any combination of the 8 line drawings.

The divergence is defined by

$$I(P \| Q) = \sum_j p_j \log \frac{p_j}{q_j}.$$

where P and Q are frequency distributions and $\sum_j p_j = \sum_j q_j = 1$. A convex set E is defined by

$$E = \{ \sum_i \lambda_i P_i \mid \sum_i \lambda_i = 1, \lambda_i \geq 0 \}.$$

Then we wish to find the universal model Q that satisfies

$$\min_Q (\max_{P \in E} I(P \| Q)).$$

It is shown that a Q exists which achieves the minimum and an algorithm is described that iteratively computes it. The changes in the divergence for this min-max universal model are shown in Table II for tree T_2.

[1] H. Freeman, "Computer processing of line drawing image," *Comput. Surveys*, vol. 6, pp. 57-97, Mar., 1974.

[2] H. Freeman, "On encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260-268, June 1961.

[3] H. Freeman, "Application of the generalized chain coding scheme to map data processing," *Proc. IEEE Comput. Soc. Conf. Pattern Recogn. Image Processing*, Chicago, IL, pp. 220-226, May 1978.

[4] Y. Yasuda and T. Kimoto, "Efficient differential chain coding for line drawings processed by Hilditch's thinning method," *Proc. Int. Conf. Commun.*, paper 2G.1, June 1982.

[5] T. Kaneko and M. Okudaira, "Encoding of arbitrary curves based on the chain code representation," *IEEE Trans. Commun.*, vol. COM-33, pp. 697-707, July, 1985.

[6] R. Hamming, *Coding and Information Theory*. Prentice-Hall, 1979.

[7] J. Dunham and C. Lu, "Highly efficient Markov coding schemes based on the chain code representation," *Proceeding of IEEE Annual Conf. on SMC*, Oct., 1987.

[8] J. Rissanen, "Arithmetic coding of strings," IBM Res. Rep. RJ 1591, June, 1975.

[9] G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Dev.*, vol. 28, no. 2, pp. 135-148, March, 1984.

[10] S. Elnahas and J. Dunham, "Entropy coding for low-bit-rate visual telecommunications" *IEEE Journal on Selected Areas in Commun.*, vol. SAC-5, no. 7, pp. 1175-83, Aug., 1987.

[11] I. Csizar, "I-divergence geometry of probability distributions and minimization problems", *Ann. of Prob.*, vol. 13, no. 1, pp. 146-158, 1975.

Figs	No. of Links	Tree T_1			Tree T_2			Kaneko & Okudaira
		Entropy	Conditioning Inaccuracy	Code Rate	Entropy	Conditioning Inaccuracy	Code Rate	
(a)	2479	1.72	1.75	1.77	1.65	1.70	1.72	1.88
(b)	1834	1.71	1.73	1.75	1.68	1.72	1.74	1.76
(c)	1307	1.28	1.40	1.43	1.18	1.33	1.34	1.38
(d)	1826	1.65	1.67	1.70	1.59	1.64	1.66	1.79
(e)	730	1.91	1.97	2.06	1.74	1.96	1.97	2.12
(f)	1967	1.79	1.81	1.82	1.73	1.81	1.82	1.99
(g)	1112	1.78	1.79	1.84	1.71	1.79	1.81	1.93
(h)	1710	1.65	1.69	1.72	1.63	1.68	1.69	1.88

Table I. Comparisons of Code Rate and Performance Using Average Relative Frequency.

Figs	No. Links	Divergence Average Model	Divergence Min-Max Model	Lambda Weights
(a)	2479	0.05	0.0659	0.0000334419
(b)	1834	0.04	0.0659	0.0006124128
(c)	1307	0.15	0.0659	0.0120001720
(d)	1830	0.05	0.0271	0.0000000126
(e)	730	0.22	0.0659	0.7277953015
(f)	1967	0.08	0.0659	0.2481296198
(g)	1112	0.08	0.0649	0.0114290086
(h)	1707	0.05	0.0643	0.0000000307

Table II. Performance of Tree T_2 with Average and Min-Max Universal Models.

Decomposition of a Multiple Access Network into a Number of Interconnected Subnetworks

Minas A. Karatzoglu and Anthony Ephremides
Department of Electrical Engineering
University of Maryland
College Park, MD 20742

Abstract

A random access system with a finite number of buffered terminals is considered. The problem of how to decompose this system into a number of interconnected subsystems in order to maximize its throughput is studied. Three different interconnection topologies are considered, a directional ring topology, a fully connected network of subsystems and a bidirectional tandem network of subsystems. For the first two topologies the problem of maximizing the throughput is reduced to a simple maximization problem, while for the last one the results are of tentative nature. The results obtained for the first two topologies are numerically evaluated for the case of a discrete time, slotted ALOHA protocol, operating on a multiaccess channel with a finite number of buffered and symmetric terminals.

1 Introduction

Given a set of terminals that must communicate over shared (multiple access) channels, the question arises whether to organize this set as interconnected subsets, each of which operates as a single multiple access system. One option is to allow the entire set of terminals to access a single common channel according to some protocol (such as ALOHA, CSMA or other similar ones). The other option is to separate the set into groups, each of which may operate as a single system under the same protocol, while intergroup traffic can be accommodated via dedicated links between the subsets. In [1] it was shown that by splitting a multiple access system with an infinite number of unbuffered users and operating under the CSMA protocol into two subnetworks we can increase the maximum stable throughput. In this paper we explore this question further by considering to what degree further subdivision improves the maximum stable throughput. Our attention is focused on multiple access channels with finite number of buffered terminals operating under a variety of protocols.

Clearly, as soon as we subdivide the set of terminals into more than two subsets, the question arises how to interconnect these subsets, i. e. the question of topological connectivity. Depending on the choice of such topology the effect of frequency reuse, that the subdivision principle ensures, will be mitigated, differently for each topology, by the need for routing multi–hop traffic.

In the following, after a precise definition of the model, we consider three different topologies and study the trade-off involved in the decomposition.

2 The Model

We assume a number of M users that can be connected by means of a single multiple access channel of given overall bandwidth W (packets/sec). We furthermore assume that the users are identical in the sense that each of them generates packets of constant bit length X at an average rate λ (packets/sec) and that the destination of the packets is uniformly distributed among the users, i. e. each packet is intended for any other user with the same probability of $1/(M-1)$. At this point nothing is assumed about what protocol is used, other than that it has a known maximum stable throughput [1] given by: $S_p(\omega, n)$ packets/packet-length , where ω is the used bandwidth in and n is the number of users. $S_p(\omega, n)$ is asummed to be a continuous and increasing function of ω and a decreasing function of n.

We consider a partition of the M users into k subsets, each with N_i, users such that $N_1 \geq N_2 \geq \ldots \geq N_k$ (unless otherwise specified) and, of course, $\sum_i N_i = M$. Communication within each subset is accomplished by means of the same protocol used for the baseline system. Communication among different subsets is achieved by means of special purpose nodes, the bridge nodes. We assume that there are two bridge nodes in each subsystem, namely the Receiving Bridge Node (RBN), which receives packets transmitted from neighboring subnetworks and broadcasts them to the users within the subnetwork or to the Transmitting Bridge Node, and the Transmitting Bridge Node (TBN), which receives packets broadcasted from within the subnetwork or from the subnetwork's RBN, that are intended for users in other subnetworks, and broadcasts them to the RBN s of neighboring subsystems.

With respect to the separation of the subsystems there are two possibilities, logical separation or physical separation. In the first case the separation of the sub–networks is achieved by bandwidth–assignment, i. e. each subnetwork uses a distinct portion of the overall bandwidth. In the second case the separation of the sub–networks is achieved by space–separation. In the case for example of Packet Radio Networks such a separation can be achieved by geographical dispersion and transmission powers control while for the case of cable LAN s by allocating a different cable channel to each LAN.

Following a construction similar to the one in [1], we consider three queueing systems that can describe the network of each subgroup of users:

1. The system with the combined aggregate queue $Q_D(i)$ of the packets in the buffers of all users in LAN i (this is a conceptual "queue" and consists of the packets generated by the users for transmission elsewhere). We denote its arrival rate by $A_D(i)$. It is allocated a portion $u_i W$ of the overall bandwidth for "service".

---

[1] We use the standard definition of maximum stable throughput as the supremum of the packet arrival rates for which the system is stable

2. The system with the queue $Q_T(i)$ of packets in the buffer of the TBN in subsystem i. We denote its arrival rate by $A_T(i)$. It is allocated a portion x_iW of the overall bandwidth for "service".

3. The system with the queue $Q_R(i)$ of packets in the buffer of the RBN in subsystem i. We denote its arrival rate by $A_R(i)$. It is allocated a portion v_iW of the overall bandwidth for "service".

In general these queueing systems are not "decoupled" and do not operate independently of one another. For each case we obtain precise relationships among them.

As a measure of the performance of the system we use the maximum stable throughput (MST), that we denote by S.

3 The Directional Ring Topology (DRT)

The first topology to be considered is the Directional Ring Topology, in which any RBN hears transmissions from the TBN of the previous subnetwork only (with subnetwork k preceding subnetwork 1). Let $n_1, n_2, \ldots, n_k$ be the order of the subnetworks in the ring. We assume that the transmission ranges within a subnetwork and between two successive subnetworks on the ring are such that the TBN of subnetwork n_i can be heard by nodes only in subnetworks n_1, n_{i+1} and n_{i-1} (but not by the RBN in subnetwork n_{i-1}). In order to avoid undesired interference we require that the bandwidths $u_{n_i}W$, $v_{n_i}W$, $x_{n_i}W$, $x_{n_{i-1}}W$ do not overlap i. e.

$$\begin{aligned} u_{n_i} + v_{n_i} + x_{n_i} + x_{n_{i-1}} &\leq 1 \quad \text{for i = 2 , ..., M} \\ u_{n_1} + v_{n_1} + x_{n_1} + x_{n_k} &\leq 1 \end{aligned} \tag{1}$$

For our purposes the division of the system is completely specified by $(k, \mathbf{N}, \mathbf{u}, \mathbf{v}, \mathbf{x})$ and our objective is to choose those parameters so that the MST of the system is maximized. Note that $\mathbf{u}, \mathbf{v}, \mathbf{x}$ denote the vectors whose components are the u_i's, v_i's and x_i's respectively and $\mathbf{N}$ denotes the vector $(N_1, \ldots, N_k)$.

We start by deriving an expression for the MST of the system. In [3] it was shown that the following inequalities hold for all i.

$$A_R(i) \leq \frac{\lambda}{2(M-1)}(M^2 - \sum_i N_i^2)$$

$$A_T(i) \leq \frac{\lambda}{2(M-1)}(M^2 - \sum_i N_i^2)$$

and that they are satisfied with equality if the system is stable.

Then it can be shown (see [3]) that the MST of the system is given by

$$S = \min_i \; \{ u_iW\frac{M}{N_i}S_p(u_iW, N_i) \;,\; MG(\mathbf{N})v_iW \;,\; MG(\mathbf{N})x_iW \}$$

where $G(\mathbf{N}) \triangleq 2(M-1)/(M^2 - \sum_j N_j^2)$.

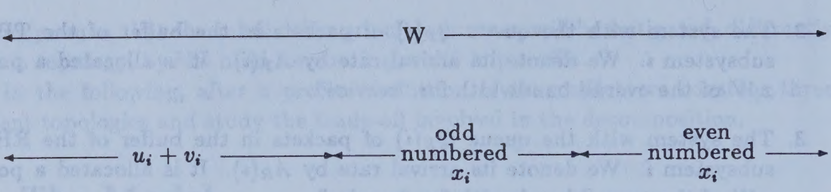


Figure 1: Bandwidth allocation for the DRT (W is in packets / sec.)

In view of the need to satisfy inequality (1) with equality as well as of the fact that all TBN s carry the same amount of traffic, the bandwidth allocation of Fig. 1 is suggested. Such an allocation however requires an even number, k, of subnetworks, since for an odd number $x_{n_k}W$ will overlap with $x_{n_1}W$. Since furthermore such an allocation simplifies the analysis we restrict our attention to even values of k . In fact it may be that k being even is a necessary condition for optimality; we do not have however a proof for this assertion yet.

We can now formulate our objective in terms of a maximization problem, namely

$$\text{(P. 1)} \qquad \max_{(k,\mathbf{N},\mathbf{u},\mathbf{v},\mathbf{x})} S \quad \text{subject to} \;\; (1.) \; \sum_i N_i = M \; ; \;\; (2.) \; u_i, v_i, x_i \in [0,1] \; ;$$

$$(3.) \;\; \text{inequality (1) is satisfied} \; ; \;\; (4.) \; k \text{ is even} \; ;$$

$$(5.) \; N_1 \geq N_2 \geq \ldots \geq N_k$$

As it was shown in [3] the following two conditions can be added to the restrictions of Problem (P. 1) without changing the solution

$$u_i = u \; ; \;\; v_i = x_i = x \; ; \;\; u + 3x = 1$$

$$uW \frac{M}{N_1} S_p(uW, N_1) = MG(\mathrm{N}) \frac{1-u}{3} W$$

The vector $\mathbf{N}$ that solves the resulting problem can be shown to satisfy the condition

$$N_1 = \ldots = N_{k-1} \geq N_k$$

Note : As a result of this last condition k is completely determined by N_1 . Therefore it might turn out to be odd. In that case we achieve an even number of subnetworks by the following procedure: Let n be the remainder of the division of M by N_1. We distinguish two cases:

(a) $n > 1$: We split the n users into two sets, one consisting of $n-1$ users and one consisting of 1 user.

(b) $n = 1$: We split the N_1 users of one of the other subnetworks as above.

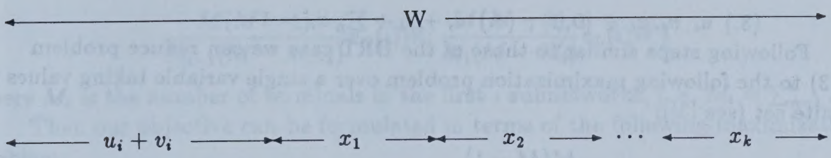


Figure 2: Bandwidth allocation for FCN (W is in packets / sec.)

Then the original problem reduces to

$$\text{(P. 2)} \qquad \max_{N_1 \in \{1,\ldots,M\}} \; uW\frac{M}{N_1}S_p(uW,N_1) \quad \text{subject to} \;\; (1.)\; u \in (0,1]\;;$$
$$(2.)\; uW \cdot (M/N_1) \cdot S_p(uW,N_1) = (M/3)\cdot G(\mathrm{N})(1-u)W\;;$$
$$(3.)\;\mathrm{N}\text{ constructed as above}$$

which is a constrained maximization problem over one variable that takes values in a finite set. To completely characterize the solution we obviously need to know the function $S_P(\cdot,\cdot)$.

4 The Fully Connected Network (FCN)

We now consider a Fully Connected topology of subnetworks, in which every TBN can be heard by the RBN s of all other subnetworks. In this case communication between any two subsystems is accomplished in a single hop (in contrast to the case of the DRT). In order to avoid undesirable interference we require that u_iW and v_iW do not overlap with any of the x_iW (for $i = 1,\ldots,k$). Furthermore no two x_jW may overlap. (Fig. 2) The following condition is necessary for such a bandwidth allocation:

$$u_i + v_i + \sum_n x_n \leq 1 \tag{2}$$

Obviously superior throughput can be achieved if inequality (2) is satisfied with equality.

Following the same procedure as in the case of the DRT we can express the MST of the FCN as

$$S = \min_i \;\{ u_iW\frac{M}{N_i}S_p(u_iW,N_i)\;,\; \frac{M(M-1)}{N_i(M-N_i)}v_iW\;,\; \frac{M(M-1)}{N_i(M-N_i)}x_iW\,\}$$

Then our objective can be formulated in terms of the following optimization problem:

$$\text{(P. 3)} \qquad \max_{(k,\mathrm{N},\mathrm{u},\mathrm{v},\mathrm{x})} S \quad \text{subject to} \;\; (1.)\; \sum_i N_i = M\;;\; (2.)\; N_1 \geq N_2 \geq \ldots \geq N_k\;;$$

$$(3.)\ u_i, v_i, x_i \in [0,1]\ ;\ (4.)\ u_i + v_i + \textstyle\sum_n x_n = 1$$

Following steps similar to those of the DRT case we can reduce problem (P. 3) to the following maximization problem over a single variable taking values in a finite set (see [3]):

$$\text{(P. 4)} \qquad \max_{N_1 \in \{1,\ldots,M-1\}} \frac{M(M-1)}{M^2 - \sum_n N_n^2} xW \text{ where : } (1.)\ N_1 = \ldots = N_{k-1} \geq N_k$$

(2.) x is the unique solution in $[\,0, 1/\phi_1]$ of the equation

$$\frac{M(M-1)}{M^2 - \sum_n N_n^2} xW = (1 - \phi_1 x) W \frac{M}{N_1} S_p(W(1 - \phi_1 x), N_1)$$

$$\text{where } \phi_i = 1 + (N_i(M - N_i))\ (M^2 - \textstyle\sum_n N_n^2)$$

At this point again knowledge of $S_p(\cdot,\cdot)$ is obviously required.

Under the reasonable assumption $S_p(\omega, 1) \equiv 1$ packets/packet-length it can be shown (see [3]) that $S(N_1 = 1) = (M/(M+2)) \cdot W$ packets/sec ; hence

$$\lim_{M \to \infty} S(N_1 = 1) = W \text{ packets/sec}$$

which implies that as M becomes large enough the optimal choice of N_1 is 1, since MST of W pack/sec is the supremum to all possible MST s. Hence it is obvious that when considering only the MST as performance measure and for large M the performance of the FCN is superior to the performance of any other topology, which is not surprising, since then the system becomes a single FDMA–like set of users for which the maximum stable throughput is known to be equal to 1.

5 Tandem Network of Subnetworks (BDTM)

We now consider a topology of a Bidirectional Tandem Network of subnetworks, in which the subnetworks are connected in tandem and the packets propagate in both directions. As in the previous cases we partition the available bandwidth into different channels. All subnetworks, except the ones at the two ends of the chain, require two TBN s and two RBN s, one for each direction. We denote them by $TBN_L(i), TBN_R(i)$ and $RBN_L(i), RBN_R(i)$, where the subscripts L, R indicate the direction of the packet propagation (L for leftward and R for rightward). The channels allocated to these queues are accordingly denoted by $v_L(i)$, $v_R(i)$, $x_L(i)W$, $x_R(i)W$. In this topology we denote by i the i-th subnetwork, starting from the left. Thus we drop the assumption $N_1 \geq N_2 \geq \ldots \geq N_k$, which would be arbitrary and might lead to a suboptimal solution. To avoid undesirable interference we require that u_iW, $v_L(i)W$, $v_R(i)W$, $x_L(i)W$, $x_R(i)W$, $x_L(i+1)W$, $x_R(i-1)W$ do not overlap, i. e.

$$u_i + v_L(i) + v_R(i) + x_L(i) + x_R(i) + x_L(i+1) + x_R(i-1) \leq 1 \qquad (3)$$

Following a procedure similar to the one for the DRT case we can derive an expression for the MST of the BDTM.

$$S = \min_i\ \{u_i W \frac{M}{N_i} S_p(u_i W, N_i)\ ;\ \frac{M(M-1)}{M_{i-1}(M - M_{i-1})} x_L(i) W ; \frac{M(M-1)}{M_i(M - M_i)} x_R(i) W ;$$

$$\frac{M(M-1)}{M_{i-1}(M-M_{i-1})} v_R(i)W; \frac{M(M-1)}{M_i(M-M_i)} v_L(i)W\}$$

where M_i is the number of terminals in the first i subnetworks, i. e. $M_i = \sum_{n=1}^{i} N_i$.

Then our objective can be formulated in terms of the following maximization problem:

(P. 5) $\quad \max S \quad$ subject to (1.) $u_i, v_L(i), v_R(i), x_L(i), x_R(i) \in [0,1]$;

(2.) $\sum_n N_n = M$; (3.) inequality (3) is satisfied

Following logical steps similar to the ones for the DRT case problem (P. 5) reduces to:

(P. 6) $\quad \max S \quad$ subject to (1.) $\sum_n N_n = M$; (2.) $u_i, v_i, x_i \in [0,1]$;

(3.) $u_i + v_i(1 + [M_{i-1}(M-M_{i-1})/M_i(M-M_i)]) + 2(x_i + x_{i-1}) = 1$;
(4.) $v_i \geq x_i$;
(5.) $u_iW \cdot (M/N_i) \cdot S_p(u_iW, N_i) \geq x_iW \cdot [M(M-1)/M_i(M-M_i)]$

where S is now given by

$$S = \min_i \; x_iW \cdot [M(M-1)/M_i(M-M_i)]$$

Unfortunately we do not yet see how to solve (P. 6), either analytically or numerically.

6 Application: Symmetric Slotted ALOHA

We apply the previous results to the case of a symmetric slotted ALOHA protocol, in which each user transmits with the same probability p, which is optimally chosen according to the number n of terminals, that share this protocol, i. e. $p = 1/n$. The MST of this protocol can be easily calculated to be:

$$S_p(\omega, n) = (1 - \frac{1}{n})^{n-1} \text{ packets/packet-length}$$

We assume a number of M terminals communicating as described in section two over a multiple access channel of total bandwidth 1 packets/packet-length. We only consider the first two topologies, i. e. the DRT and the FCN. For the latter it can be easily shown that for all M the unique optimal number of subnetworks is given by $k_{FCN} = M$. For the former the optimal number of subnetworks is shown in Fig. 4 and 5 as a function of M. The actual value of the optimal MST for these two topologies as well as of the baseline system is shown in Fig. 3 as a function of M. The asymptotic behavior of these systems can also be seen in Fig, 3. As expected $\lim_{M\to\infty} S_{FCN} = 1$ and $\lim_{M\to\infty} S_{BLS} = 1/e$. We can also see (it can actually be proved) that for this protocol $\lim_{M\to\infty} S_{DRT} = 2/3$.

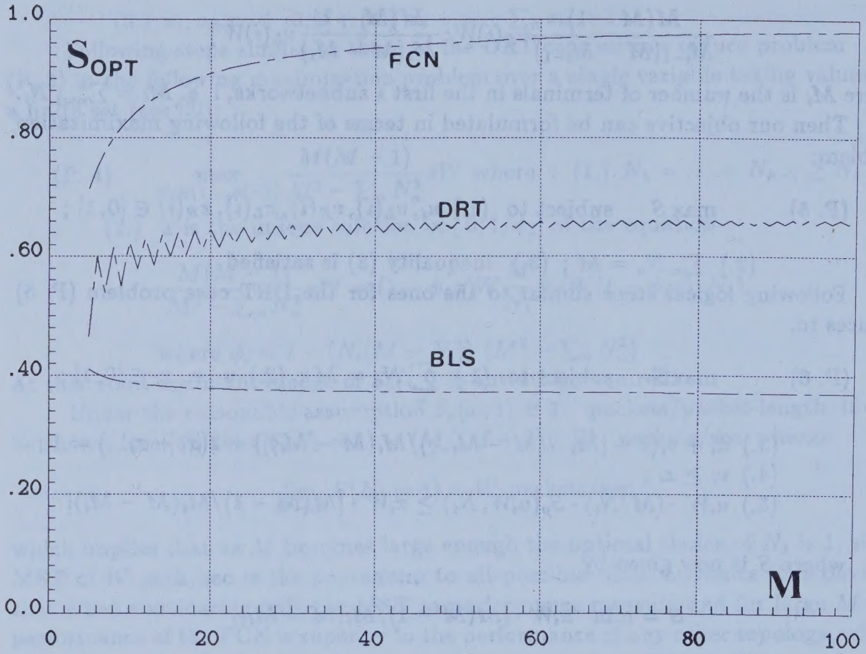


Figure 3: Optimal MST for Directional Ring Topology (DRT), Fully Connected Network (FCN) and the Baseline system (BLS)

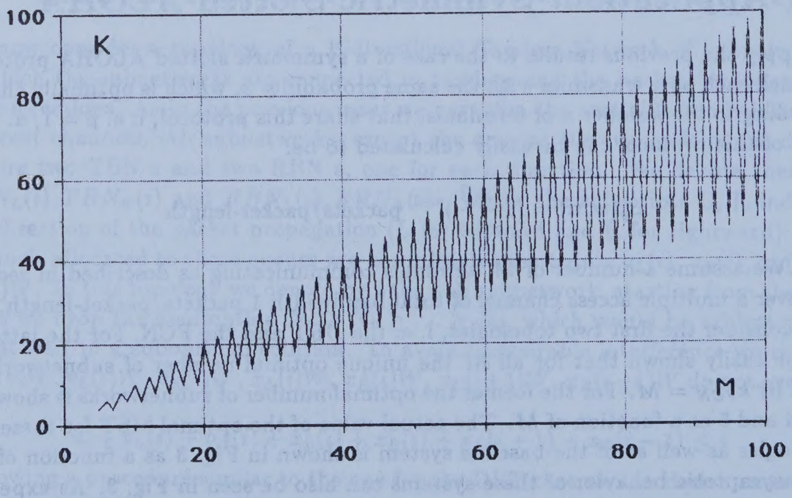


Figure 4: Optimal number of subnetworks for the DRT

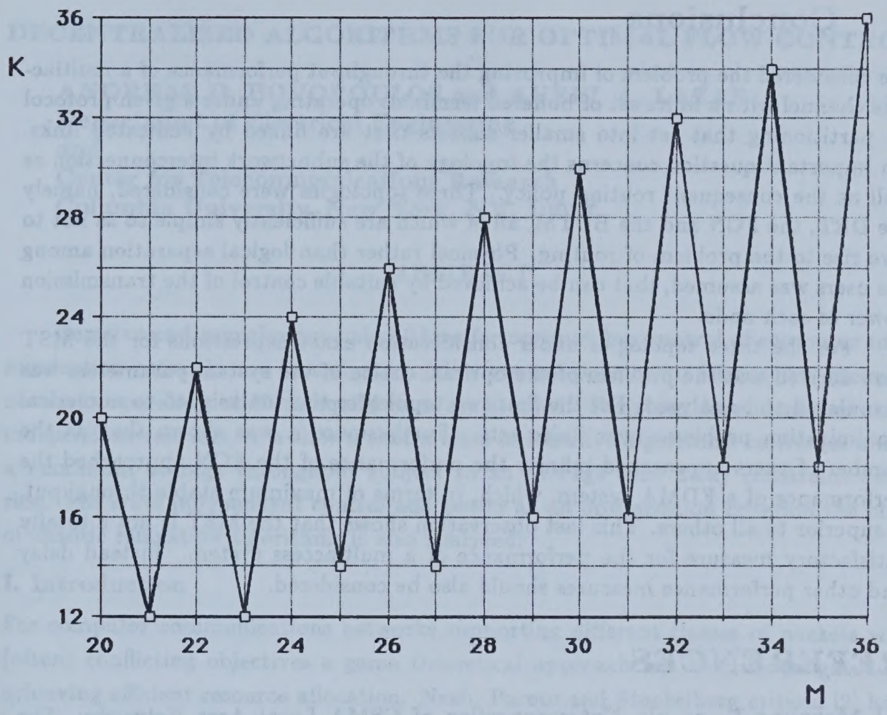


Figure 5: Detailed view of the optimal number of subnetworks for the DRT

7 Conclusions

We considered the problem of improving the throughput performance of a multiaccess channel with a finite set of buffered terminals operating under a given protocol by partitioning that set into smaller subsets that are linked by dedicated links. An important question concerns the topology of the subnetwork interconnection as well as the consequent routing policy. Three topologies were considered, namely the DRT, the FCN and the BDTM, all of which are sufficiently simple so as not to give rise to the problem of routing. Physical rather than logical separation among the users was assumed, that can be achieved by suitable control of the transmission power of each node.

For the three topologies under consideration exact expressions for the MST were derived and the problem of the optimal choice of the system parameters was formulated and analyzed. For the first two topologies the analysis led to numerical maximization problems over finite sets. Furthermore it was shown that as the number of users approached infinity the performance of the FCN approached the performance of a FDMA system, which, in terms of maximum stable throughput, is superior to all others. This last observation shows that the MST is not a totally satisfactory measure for the performance of a multiaccess system. Instead delay and other performance measures should also be considered.

REFERENCES

[1] **Merakos L. et al**: "Interconnection of CSMA Local Area Networks: The Frequency Division Approach" *Transactions on Communications, vol. COM-35, No. 7, pp 730 - 738, July 1987*

[2] **Loynes R. M.** : "The Stability of a Queue with non–Independent Interarrival and Service Times" *Cambridge Philosophical Society ; vol. 58 ; pp 497 - 520; 1962*

[3] **Karatzoglu M. A.** : " On the Stability and Performance of Multiple Access Channels with a Finite Number of Buffered Terminals" *M. S. Thesis, Department of Electrical Engineering, University of Maryland*

DECENTRALIZED ALGORITHMS FOR OPTIMAL FLOW CONTROL

ANDREAS D. BOVOPOULOS and AUREL A. LAZAR
Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY, 10027

ABSTRACT

Iterative and asynchronous algorithms for optimal flow control of computer communication networks are investigated. Flow control policies for Markovian queueing networks operating at Nash equilibrium are studied. For state dependent and state independent controls, it is shown that a class of iterative algorithms converges under a maximum average throughput subject to an average time delay constraint criterion. For state independent control and power as an optimization criterion the class of chaotic relaxation algorithms is also analyzed.

I. Introduction

For computer communications networks supporting different classes of packets with (often) conflicting objectives a game theoretical approach has been investigated for achieving efficient resource allocation. Nash, Pareto and Stackelberg criteria [2] have been studied [10], [13], [7], [9].

Hsiao and Lazar [10], [13] have developed a framework for decentralized optimal flow control for Markovian queueing networks. They have shown that, using an average network throughput time delay trade off, the Nash equilibrium (if it exists) is achieved using a bang-bang (window) flow control policy. The control policy is based on a decentralized information structure, *i.e.* local information. Finding the Nash equilibrium point is equivalent to computing a fixed point. Hsiao and Lazar also proposed a decentralized algorithm for determining the optimal flow control of a queueing system having an exponential server [10]. No convergence properties have however been given.

The nature of the algorithm studied in [10] is based upon an iterative procedure whose formalism has been advanced by Jaffe [14]. Within this, Barath-Kummar and Jaffe [1] have advanced heuristic algorithms with desirable properties in a decentralized networking environment. One such algorithm, called greedy, has been proven by Douligeris and Mazumdar [9] to converge for a M/M/1 queueing model with *two user* classes to the Nash equilibrium point.

In this paper Jaffe's iterative model is extended to include asynchronous algorithms [8]. In this framework each user updates his flow control policy asynchronously, that is, a particular user might change his flow control policy even if it has not obtained an updated version of the flow control policies from the other users. This paradigm is borrowed from previous work on decentralized computation [8], [3], [4], [17].

This paper is organized as follows. In section II the statement of the problem is presented. A class of iterative algorithms that converge to the Nash equilibrium point of a BCMP network with two user classes is studied in section IV. The average throughput is the utility function of each user; the expected time delay does not exceed a given upper bound. Properties that guaranty convergence are found in section III. Asynchronous algorithms are introduced in section V.

II. The Statement of the Problem

A BCMP network with M exponential servers, is shared by K different classes of packets that are processed according to the FIFO policy. Let μ^i be the service rate of the i^{th} server, and $\mathbf{M}$ be the $1 \times M$ matrix, $[\mu^1 \cdots \mu^M]$. Let $\mathbf{R}^k = [r^{kij}]$ be the $M \times M$ routing matrix of the k^{th} class of packets $(1 \leq k \leq K,\ 1 \leq i \leq M,\ 1 \leq j \leq M)$. In this notation, class k packets are routed from node i to node j with probability r^{kij}, and join node j with probability $r^{k \cdot j}$. $\mathbf{\Lambda} = [\lambda^1_{j_1} \cdots \lambda^K_{j_K}]$ represents the set of arrival rates of the K packet classes; it also denotes the set of controls. Thus, user k operates either with a state dependent Poisson arrival rate $\lambda^k_{j_k}$, (j_k is the number of packets that have not yet been acknowledged), or with a state independent Poisson arrival rate λ^k. To determine the set of controls, two optimality criteria are considered. These criteria are based upon the average throughput $E\gamma^k$ and the expected time delay $E\tau^k$ and reflect the trade off between the two [15], [11], [12]. With respect to these criteria users operate at a Nash equilibrium point. Intuitively, a set of users operate at a Nash equilibrium if, given the strategies of the other users, none of them has an incentive to change unilaterally its operating policy [2], [10], [13].

In order to achieve the Nash equilibrium two classes of distributed flow control algorithms are investigated. In the first class, users attempt to reach the Nash operating point using a Gauss-Seidel type iterative procedure. The algorithm requires that one user changes his flow control policy at a time. After adjusting its policy that maximizes the user's average throughput under its time delay constraint, the value of the new policy is broadcasted to all the other users. This step by step iterative procedure results in the optimal operating point for the network. Formally, the class of iterative algorithms above is studied with respect to the user optimization based upon an average throughput subject to a bounded expected time delay constraint criterion. Thus user k, $1 \leq k \leq K$, attempts to

$$\max_{E\tau^k \leq T^k} E\gamma^k \quad ,$$

where T^k is the upper bound on the expected time delay for the k^{th} class of packets [11], [12].

The second class of decentralized algorithms investigated here is borrowed from the field of distributed computation. In this case, each user will recompute its flow control policy asynchronously. The computation of a new flow control policy takes place even if the latest changes in the control strategies of the other users are not available. However, the policies of other users will be available within a bounded time. The class of asynchronous algorithms is investigated with respect to a power based user optimization criterion. Power based user optimization calls each user k for maximizing the ratio of the weighted average throughput over the expected time delay:

$$\frac{(E\gamma^k)^{\beta_k}}{E\tau^k} \quad .$$

The parameter β_k may be adjusted to achieve different trade off points between average throughput and expected time delay.

In order to prove convergence of iterative and asynchronous algorithms, contraction mapping type arguments are invoked. These often require proving monotonicity properties. A set of useful monotonicity properties for BCMP networks are given in the next section.

III. Monotonicity Properties of BCMP Networks

Two monotonicity results are stressed in this section. In the first, it is shown that the expected time delay of the k'th class of packets in a BCMP network is a increasing function of the control of the individual users. Under certain conditions this function is convex increasing (see Proposition 1). In the second, a BCMP network with two user classes is considered. Results regarding monotonicity properties of the average throughput and the expected time delay with respect to the individual controls of the other users are given in Proposition 2.

Let the $1 \times M$ matrix $\mathbf{\Theta}^k = [\theta^{k1} \cdots \theta^{kM}]$ be the solution of the traffic flow equations for the k^{th} class of packets of the BCMP network defined in the previous section, *i.e.*

$$\mathbf{\Theta}^k = \mathbf{\Lambda}^k + (\mathbf{\Theta}^k \wedge \mathbf{M})\mathbf{R}^k \quad .$$

Here $\mathbf{\Lambda}^k$ denotes the load vector of the input traffic flows of the k^{th} traffic class:

$$\mathbf{\Lambda}^k = \lambda^k [r^{k\cdot 1} \cdots r^{k\cdot M}] \quad ,$$

for all k, $k = 1, 2, \cdots, K$. If the BCMP network is stable:

$$\mathbf{\Theta}^k = \mathbf{\Lambda}^k + \mathbf{\Theta}^k \mathbf{R}^k \quad ,$$

or

$$\mathbf{\Theta}^k = \mathbf{\Lambda}^k (\mathbf{I} - \mathbf{R}^k)^{-1} \quad .$$

With

$$\alpha^k = [\alpha^{k1} \cdots \alpha^{kM}] \overset{\text{def}}{=} [r^{k\cdot 1} \cdots r^{k\cdot M}](\mathbf{I} - \mathbf{R}^k)^{-1} \quad ,$$

we obtain

$$\theta^{kj} = \lambda^k \alpha^{kj} \quad .$$

Thus the expected time delay of user k packets in the network is given by [5]

$$E\tau^k = \sum_{j=1}^{M} \frac{\alpha^{kj}}{\mu^j - \sum_{i=1}^{K} \alpha^{ij}\lambda^i} \; .$$

Proposition 1. *The expected time delay $E\tau^k$ of the k^{th} class packets of a BCMP network is a nondecreasing function of λ^i for all i, $i = 1, 2, \cdots, K$, and a convex increasing function of λ^i if $\alpha^{kj}\alpha^{ij} > 0$ for some j, $j = 1, 2, \cdots, M$.*

Proof : The proof is straightforward.

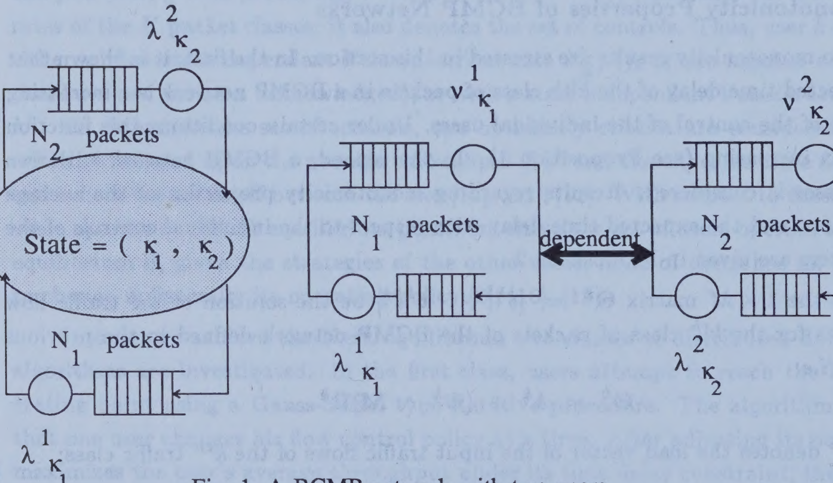

$\lambda^2_{\kappa_2}$
N_2 packets
State = (κ_1 , κ_2)
N_1 packets
$\lambda^1_{\kappa_1}$
$\nu^1_{\kappa_1}$
$\nu^2_{\kappa_2}$
N_1 packets
dependent
N_2 packets
$\lambda^1_{\kappa_1}$
$\lambda^2_{\kappa_2}$

Fig. 1: A BCMP network with two users subject to state dependent flow control, and its equivalent representation.

Two users share the resources of a BCMP network (see Figure 1). The following result holds.

Proposition 2. *For all k, j, $k, j = 1, 2$ and $k \neq j$,*

(i) the average throughput $E\gamma^k$ and the expected time delay $E\tau^k$ are increasing functions of λ^k,

(ii) the average throughput $E\gamma^j$ and the expected time delay $E\tau^j$ are decreasing and increasing functions of λ^k, respectively.

Proof : See [5].

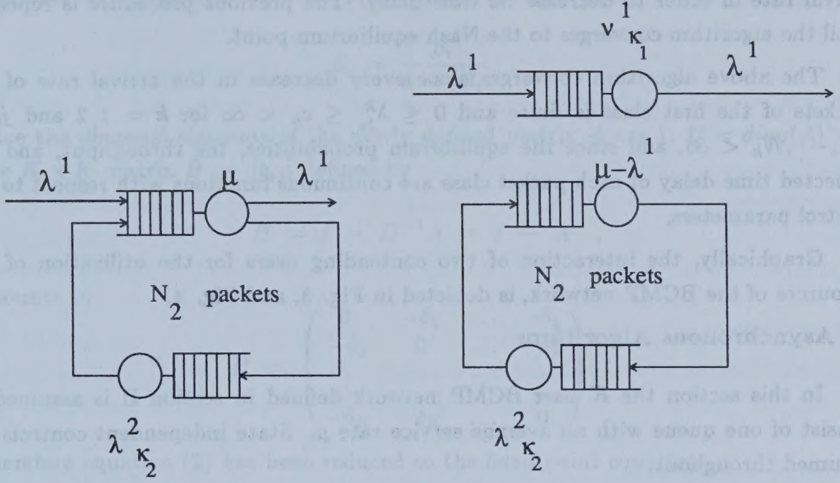


Fig. 2: Network subject to state dependent and state independent flow control, and its equivalent networks.

For the network depicted in Figure 2 the first class of packets is subject to state independent intensity control, and the second class of packets is subject to state dependent intensity control. As shown in [5], Proposition 2 still holds (in this case $N_1 \to \infty$). Thus BCMP networks with state dependent and state independent controls have the properties exposed in Proposition 2.

IV. Iterative Algorithms

In the sequel the iterative algorithm proposed in [10] (for the BCMP network shown in Figure 1) is considered and its convergence properties demonstrated. Both state dependent and state independent controls are considered.

Let us assume that, as an initial condition, user 1 operates on the network alone. If user's 1 set of controls is state independent, the optimal arrival rate is the maximum feasible rate that does not violate the time delay constraint. If user's 1 set of controls is state dependent, the optimal flow control is the solution of a Linear Program ([6]). Based on the optimal flow control policy of user 1, the effective service rate of the second user can be evaluated by finding the Norton equivalent $\nu^2_{j_2}$ for $j_2 = 1, 2, \cdots, N_2$. If the value of $(\nu^2_1)^{-1} > T^2$, the second user cannot send any packets into the network. In this case the network operates at a Nash equilibrium point. If $(\nu^2_1)^{-1} \leq T^2$, the second user is able to send packets into the network. The optimal flow control of the second user is determined as follows. If the second user is subject to state independent intensity control its optimal arrival rate is the maximum feasible rate that does not violate its time delay constraint. According to Propositions 1 and 2, the expected time delay of the first user does not decrease. If the new time delay of the packets of the first user does not exceed the upper bound T_1 then the network operates at a

Nash equilibrium point; otherwise, owing to Propositions 1 and 2, user 1 decreases its arrival rate in order to decrease its time delay. The previous procedure is repeated until the algorithm converges to the Nash equilibrium point.

The above algorithm converges since every decrease in the arrival rate of the packets of the first class is finite and $0 \leq \lambda_{j_k}^k \leq c_k < \infty$ for $k = 1,2$ and $j_k = 0,1,\cdots,N_k < \infty$, and since the equilibrium probabilities, the throughput, and the expected time delay of each packet class are continuous functions with respect to the control parameters.

Graphically, the interaction of two contending users for the utilization of the resources of the BCMP network, is depicted in Fig. 3, and Fig. 4.

V. Asynchronous Algorithms

In this section the K user BCMP network defined in section II is assumed to consist of one queue with an average service rate μ. State independent controls are assumed throughout.

The power of user k is given by

$$(\lambda^k)^{\beta_k}(\mu - \sum_{j=1}^{K} \lambda^j) \quad .$$

Thus, a Nash equilibrium point (if it exists) corresponds to the solution of the following set of equations:

$$(1+\beta_k)\lambda^k + \sum_{j=1,\ j\neq k}^{K} \beta_j \lambda^j = \beta_k \mu \ , \tag{1}$$

for all k, $k = 1,2,\cdots,K$. Note that (1) represents a system of K linear equations with K unknowns.

Let x^T denote the vector $[\lambda^1 \ \cdots \ \lambda^K]$ and A the $K \times K$ matrix $[a_{ij}]$ with $a_{ii} = 1 + \beta_i$ and $a_{ij} = \beta_i$ for all $j, j \neq i$. Let $b^T = [b_1 \ \cdots \ b_K]$, where $b_i = \beta_i \mu$. The derivation of the Nash equilibrium of this simple network calls for the solution of the system of linear equations

$$Ax = b \ . \tag{2}$$

It is easy to prove that $det(A) = 1 + \sum_{i=1}^{K} \beta_i$. Thus *the Nash equilibrium exists and is unique.* Since $(A^{-1})_{ii} = \frac{1+\sum_{j=1, j\neq i}^{K} \beta_j}{det(A)}$, for every $1 \leq i \leq K$, and $(A^{-1})_{ij} = -\frac{\beta_i}{det(A)}$ for every $1 \leq i \neq j \leq K$, equation (2) has the following unique solution

$$x = [\beta_1\mu \ \cdots \ \beta_K\mu]^T \frac{1}{1 + \sum_{i=1}^{K} \beta_i} \quad .$$

Using a straightforward linear transformation equation (2) is reduced to $Ax = b$, where $a_{ii} = 1$ and $a_{ij} = \delta_i$ for all j, $j \neq i$, and $b_i = \delta_i \mu$ with

$$\delta_i = \frac{\beta_i}{1 + \beta_i} .$$

Since the diagonal elements of the newly defined matrix A are 1, $D = diag(A) = I$. The $K \times K$ matrix $B = [b_{ij}]$ defined by

$$B = I - D^{-1}A = I - A ,$$

amounts to

$$B = \begin{pmatrix} 0 & -\delta_1 & \dots & -\delta_1 \\ -\delta_2 & 0 & \dots & -\delta_2 \\ \vdots & \vdots & \ddots & \vdots \\ -\delta_K & -\delta_K & \dots & 0 \end{pmatrix} .$$

Therefore equation (2) has been reduced to the fixed point equation

$$x = Bx + b \quad . \tag{3}$$

Chazan and Miranker [8] considered chaotic relaxation methods for solving linear systems of equations. In the sequel these results are briefly presented and subsequently applyed to the problem above.

The computations are done in steps calles iterations. These iterations are indexed by $t = 0, 1, 2, \cdots$ and t can be viewed as a discrete time variable for the system. At time $t + 1$ a set $U(t + 1)$ of coordinates of the vector $x(t + 1)$ is updated as follows:

$$x_i(t+1) = \sum_{j=1}^{K} b_{ij} x_j(t - \tau(t,i,j)) + b_i , \quad if\ i \in U(t) \quad , \tag{4}$$

$$x_i(t+1) = x_i(t) , \quad if\ i \notin U(t) \quad ,$$

where $\{\tau(\cdot,\cdot,\cdot)\}$ are delay terms. The initial vector is $x(0)$ and $x_i(t) = x_i(0)$ for all $t < 0$. Let Γ denote a choice for $\{U(t), \tau(t,i,j),\ t \geq 0,\ 1 \leq i,j \leq K\}$. Then $x(0)$ and Γ completely define the evolution of $x(t)$, $t > 0$ (the sample path of $x(t)$, $t > 0$) given by (4).

Let $\Gamma_{d,s}$ and Γ_d correspond to the case that the (random) delays are bounded by d (*i.e.* $\tau(t,i,j) \leq d$) and each component x_i, $i = 1, \cdots, K$, of the vector x is updated at least every s steps during the iterative procedure or infinitely often, respectively. Then the following holds.

Theorem 1. *The sequence of iterates $x(t)$, $t = 0, 1, 2, \cdots$ converges for all choices of $\Gamma_{d,s}$ and Γ_d, iff $\rho(|B|) < 1$.*

Proof : For a proof and extensions see [8].

Let us now return to equation (3) above. The dominant nonnegative eigenvalue of the associated nonnegative matrix $|B|$

$$|B| \;=\; \begin{pmatrix} 0 & \delta_1 & \dots & \delta_1 \\ \delta_2 & 0 & \dots & \delta_2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_K & \delta_K & \dots & 0 \end{pmatrix}$$

is bounded by

$$\min_i \sum_{j=1}^{K} |b_{ij}| \le \rho(|B|) \le \max_i \sum_{j=1}^{K} |b_{ij}| \quad ,$$

with equality on either side implying equality throughout [18, p. 8].

Note that the minimum Σ_i of the elements of row i is

$$\Sigma_i \ge \min_i (K-1)\delta_i \;=\; (K-1)\min_i \delta_i \;=\; \frac{K-1}{2} \quad .$$

Observe that if $K \ge 3$ then $\Sigma_i \ge 1$. Therefore the chaotic iterative method described by (4) does not converge for $K \ge 3$. For $K = 2$, $\rho(|B|) \le \max_i \delta_i \;<\; 1$. Hence, we have shown the following.

Proposition 3. *For an $M/M/1$ queueing system with a user based power criterion the chaotic relaxation algorithm converges for all the choices of $\Gamma_{d,s}$ and Γ_d, only for $K = 1, 2$.*

The proof of Proposition 3, in addition verifies that a synchronous Jacobi iteration algorithm does not converge for $K \;>\; 2$.

VII. Conclusions

Decentralized algorithms for models of computer communication networks have been studied. Two classes of algorithms have been analyzed.

For the class of iterative procedures, an algorithm previously proposed by Hsiao and Lazar [10] for a BCMP network with two users has been shown to converge to the Nash equilibrium.

A class of asynchronous algorithms for optimal flow control has been also proposed. For this class, the power based user optimization criterion has been employed. It has been shown that the asynchronous version of the greedy algorithm [1, p. 429] converges only for at most two users.

Acknowledgements

We would like to thank Dr. Costas Courcoubetis of Bell Laboratories for introducing us to the theory of asynchronous algorithms.

The research reported here was supported by the National Science Foundation under Grant # CDR-84-21402.

References

[1] Barath-Kummar, K. and Jaffe, J., "A New Approach to Performance Oriented Flow Control", *IEEE Transactions on Communications,* Vol. COM-29, No. 4, April 1981, pp. 427-435.

[2] Basar, T. and Olsder, G.J., *Dynamic Noncooperative Game Theory,* Academic Press, New York, 1982.

[3] Baudet, G. M., "Asynchronous Iterative Methods for Multiprocessors", *Journal of the Association for Computing Machinery,* Vol. 25, No. 2, April 1978, pp. 226-244.

[4] Bertsekas, D.P., "Distributed Asynchronous Computation of Fixed Points", *Mathematical Programming,* Vol. 25, No. 2, April 1978, pp. 226-244.

[5] Bovopoulos, A.D. and Lazar, A.A., "Analysis and Flow Control of a Multiclass Network under Different Control Policies", *Proceedings of the First International Conference on Advances in Communication and Control Systems",* Washington, D.C., June 16-18, 1987.

[6] Bovopoulos, A.A. and Lazar, A.A., "Optimal Load Balancing for Markovian Queueing Networks." presented in part in the *Proceedings of the* 30^{th} *Midwest Symposium on Circuits and Systems,* Syracuse University, August 17-18, 1987.

[7] Cansever, D. H., "Decentralized Algorithms for Flow Control in Networks", *Proceedings of the 25th Conference in Decision and Control,* Athens, Greece, December 1986, pp. 2107-2112.

[8] Chazan, D. and Miranker, W., "Chaotic Relaxation", *Linear Algebra and its Applications,* Vol. 2, 1969, pp. 199-222.

[9] Douligeris, C. and Mazumdar, R., "An Approach to Flow Control in an Integrated Environment" *CTR Technical Report, CU-CTR-TR-50,* Columbia University, 1987.

[10] Hsiao, M.T. and Lazar, A.A., "Bottleneck Modeling and Decentralized Optimal Flow Control - II. Individual Objectives", *Proceedings of the Nineteenth Annual Conference on Information Sciences and Systems,* The Johns Hopkins University, Baltimore, MD, March 1985, pp. 558-563.

[11] Hsiao, M.T. and Lazar, A.A., "Optimal Flow Control of Multiclass Queueing Networks with Decentralized Information", *Proceedings of the INFOCOM'87,* San Francisco, CA, March 31 - April 2, 1987, pp. 652-661.

[12] Hsiao, M.T. and Lazar, A.A., "Optimal Decentralized Flow Control of Markovian Queueing Networks", *Proceedings of the Third International Conference on Data Communication Systems and their Performance,* Rio de Janeiro, Brazil, June 22-25, 1987.

[13] Hsiao, M.T. and Lazar, A.A., "A Game Theoretic Approach to Optimal Decentralized Flow Control of Markovian Queueing Networks with Multiple Controllers", *Proceedings of Performance87,* Brussels, Belgium, December 9-11, 1987.

[14] Jaffe, J.M., "Flow Control Power is Nondecentralizable", *IEEE Transactions on Communications,* Vol. COM-29, September 1981, pp. 1301-1306.

[15] Lazar, A. A., "Optimal Flow Control of a Class of Queueing Networks in Equilibrium", *IEEE Transactions on Automatic Control,* Vol. AC-28, No. 11, November 1983, pp. 1001-1007.

[16] Luenberger, D.G., *Linear and Nonlinear Programming,* Second Edition, Addison Wesley, New York, 1984.

[17] Li, S. and Basar, T., "Asymptotic Agreement and Convergence of Asynchronous Stochastic Algorithms", *IEEE Transactions on Automatic Control,* Vol. AC-32, No. 7, July 1987, pp. 612-618.

[18] Seneta, E., *Non-negative Matrices and Markov Chains* , Second Edition, Springer-Verlag, 1981.

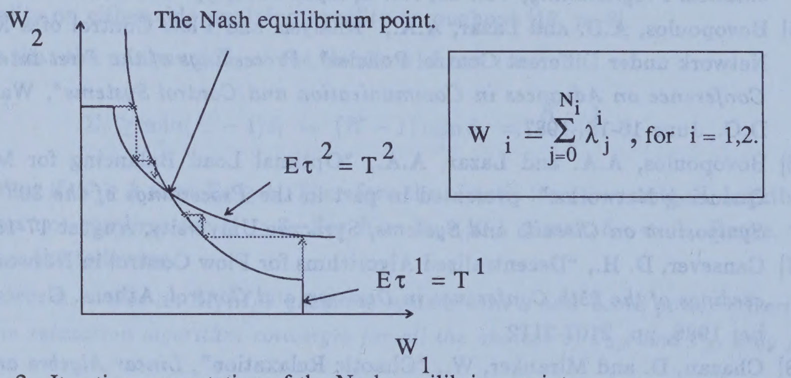

W_2
The Nash equilibrium point.
$W_i = \sum_{j=0}^{N_i} \lambda^i_j$, for i = 1,2.
$E\tau^2 = T^2$
$E\tau^1 = T^1$
W_1

Fig. 3 : Iterative computation of the Nash equilibrium point of a BCMP network with two users, subject to state dependent flow control.

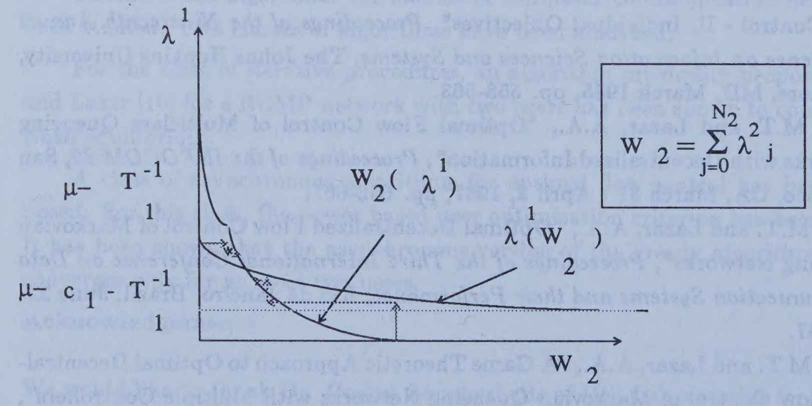

λ^1
$W_2 = \sum_{j=0}^{N_2} \lambda^2_j$
$\mu - T_1^{-1}$
$W_2(\lambda^1)$
$\lambda^1(W_2)$
$\mu - c_1 - T_1^{-1}$
W_2

Fig. 4 : Iterative computation of the Nash equilibrium of a BCMP network subject to state independent and state dependent flow control.

ON PARETO OPTIMAL FLOW CONTROL IN A MULTICLASS ENVIRONMENT

CHRISTOS DOULIGERIS and RAVI MAZUMDAR
Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY, 10027

ABSTRACT

A Pareto optimal solution to the flow control in a multiclass communications environment is defined. Several classes of customers compete for the resources provided by an exponential server, using the power as their performance criterion. Necessary and sufficient conditions for the throughputs of the different classes so as to achieve these Pareto optimal points are given. Several properties of these solutions are also discussed.

I. INTRODUCTION

Multiobjective criteria optimization problems arise in environments, where either a number of different users compete for a common resource or a single user has more than one objectives to satisfy. This type of problem is common in economic environments, as well as in technical ones. In the latter, the computer communications networks environment is an area, where either the presence of many users or the multiple criteria a single user has to satisfy and/or optimize give rise to a large number of interesting multiobjective optimization problems.

The existence of multible objectives makes the definition of an optimum solution a hard, if not an impossible task. The notion of Pareto optimality [8] provides a set of solutions, which have an intuitively satisfactory property. According to this idea in a multiobjective criterion optimization problem a solution is optimal if there is no other solution that could simultaneously improve the performance of all the objectives.

In this paper we find necessary and sufficient conditions for Pareto optimal points for flow control in a computer network setting where several classes of users compete for the resources of a given channel, represented in our system by an exponential server with service rate μ. These different classes arise in a number of different ways. The integration of voice, video and data gives three classes with different and frequently conflicting objectives. Locally generated traffic at a certain node might have different characteristics than traffic that uses this certain node as an intermediate one.

The conflicting objectives inherent in each class are incorporated in the cost criterion to be used. The power, defined as the weighted ratio of the average throughput over the average time delay, is a criterion that has received a considerable attention in the study of flow control in computer networks [1][2][4][6]. It includes the main desirable performance criteria -the throughput and the average time delay- it is analytically tractable and gives a wide range of performance criteria through the change of the weighting factor.

This paper is organized as follows. In section II the model is presented and the performance criterion is discussed. Section III presents the necessary and sufficient conditions for the Pareto optimal points, as well as an analytic discussion on the properties of the solution and section IV concludes the paper.

II. THE MODEL AND THE PERFORMANCE CRITERION

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$, be the arrival rates of n Poisson packet streams, that are serviced by a single server, on a first come first served basis, represented here by an exponential server with a negative exponential distribution with mean μ. Before joining a common queue to be served each user blocks a certain number of packets so as to achieve a desirable performance characteristic (See Fig. 1). An arriving customer to a certain class i is blocked with probability B_i, so as the arrival rate to the queue is γ_i. Note that this arrival rate is also the throughput of class i.

The maximization of the throughput and the minimization of the average time delay are the conflicting objectives of each class of users. A performance criterion to include both this criteria in a harmonious way is the power, introduced in [1] and extended in [2].

Definition 1.

Let γ_i be the throughput of class i and $D_i(\gamma_1, \ldots, \gamma_n)$ the corresponding average time delay, then the power of class i is

$$P_i(\gamma_1, \ldots, \gamma_n, r_1, \ldots, r_n) = \frac{{\gamma_i}^{r_i}}{D_i(\gamma_1, \ldots, \gamma_n)}, \tag{1}$$

where the r_i's are real positive constants.

Changing the values of the weighting factors r_i a wide range of performance criteria can be studied, where different classes are willing to sacrifice throughput for time delay and vice versa.

In order to have a stable queue the set of admissible throughputs has to be defined.

Definition 2.

A n-tuple $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ of nonnegative real numbers defines an admissible set of throughputs iff

$$\sum_{i=1}^{n} \gamma_i < \mu. \tag{2}$$

Define the following vectors

$$\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n) \qquad \text{and} \qquad \mathbf{P}(\gamma) = (P_1(\gamma), P_2(\gamma), \ldots, P_n(\gamma))$$

Our search for optimal controls is naturally restricted to the class of admissible throughputs. Let Γ be the set of all admissible sets of throughputs i.e.

$$\Gamma = \left(\gamma = (\gamma_1, \ldots, \gamma_n) : \sum_{i=1}^{n} \gamma_i < \mu, \;\; \gamma_i \geq 0 \;\; \forall \;\; 1 \leq i \leq n \right)$$

The system behaves as an M/M/1 queue with arrival rate $\gamma_{tot} = \gamma_1 + \gamma_2 + \ldots + \gamma_n$ and service rate μ. Thus the average delay is

$$D_i(\gamma) = \frac{1}{\mu - \gamma_{tot}}$$

and the power of user i is

$$P_i(\gamma_1, \gamma_2, \dots, \gamma_n) = \gamma_i^{r_i} \ (\mu - \gamma_{tot}).$$

The objective of each user is the maximization of its power. To achieve this, each user can control its throughput γ_i. But the maximization of its power conflicts with the objectives of the other users. A cooperative solution seems to be a way to the resolution of this conflict. The cooperative solution with the property, that any deviation from the optimal point will result in the worsening of the situation of at least one of the classes is known as the Pareto optimal point and is defined as follows[3].

Definition 3.

A point $\gamma^* \in \Gamma$ is called Pareto optimal iff for any $(\gamma^* + \delta\gamma) \in \Gamma$ either

$$P_i(\gamma^* + \delta\gamma) - P_i(\gamma^*) = 0 \quad \forall \quad 1 \leq i \leq n$$

or for at least one i

$$P_i(\gamma^* + \delta\gamma) - P_i(\gamma^*) < 0.$$

In the next section necessary and sufficient conditions that the throughputs of each class have to follow in order to achieve the Pareto optimal points are given.

III. THE NECESSARY AND SUFFICIENT CONDITIONS

The determinant of the Jacobian is is a key element in the study of multiobjective optimization problems. The equivalent conditions provided by the following lemma are essential for the proof of the main result of this paper.

Lemma 1.

The following conditions are equivalent:

$$(i) \quad \det \frac{\vartheta \mathbf{P}}{\vartheta \gamma} = 0 \tag{3}$$

$$(ii) \quad \left(\prod_{i=1}^{n} r_i \left(\mu - \sum_{k=1}^{n} \gamma_k \right) \right)^{n-1} \left(\sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} r_k (r_i + 1)\gamma_i - \prod_{k=1}^{n} r_k \mu \right) = 0. \tag{4}$$

Proof :

$$\det \frac{\vartheta \mathbf{P}}{\vartheta \gamma} = 0 \Leftrightarrow \begin{vmatrix} \frac{\vartheta P_1}{\vartheta \gamma_1} & \frac{\vartheta P_1}{\vartheta \gamma_2} & \cdots & \frac{\vartheta P_1}{\vartheta \gamma_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\vartheta P_n}{\vartheta \gamma_1} & \frac{\vartheta P_n}{\vartheta \gamma_2} & \cdots & \frac{\vartheta P_n}{\vartheta \gamma_n} \end{vmatrix} = 0$$

Equivalently the following determinant is equal to 0.

$$\begin{vmatrix} \gamma_1^{r_1-1}(\mu r_1 - r_1 \sum_{k=1}^{n} \gamma_k - \gamma_1) & \cdot & -\gamma_1^{r_1} & \cdot & -\gamma_1^{r_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\gamma_i^{r_i} & \cdot & \gamma_i^{r_i-1}(\mu r_i - r_i \sum_{k=1}^{n} \gamma_k - \gamma_i) & \cdot & -\gamma_i^{r_i} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\gamma_n^{r_n} & \cdot & -\gamma_n^{r_n} & \cdot & \gamma_n^{r_n-1}(\mu r_n - r_n \sum_{k=1}^{n} \gamma_k - \gamma_n) \end{vmatrix}$$

We now do the following calculations:

i) Divide each row i by $\gamma_i^{r_i}$

ii) Substitute row n by the sum of rows 1 to n where row i has been multiplied by $\prod_{\substack{k=1 \\ k \neq i}}^{n} r_i$

and have

$$\det \frac{\vartheta \mathbf{P}}{\vartheta \gamma} = 0 \Leftrightarrow \left[\sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} r_k (r_i + 1) \gamma_i - \prod_{k=1}^{n} r_k \mu \right] |A| = 0.$$

where

$$|A| = \begin{vmatrix} r_1 \mu - r_1 \sum_{k=1}^{n} \gamma_k - \gamma_1 & \cdots & -\gamma_1 & \cdots & -\gamma_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ -\gamma_i & \cdots & r_i \mu - r_i \sum_{k=1}^{n} \gamma_k - \gamma_i & \cdots & -\gamma_i \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & \cdots & 1 \end{vmatrix}$$

The above determinant is of the form

$$|A| = \begin{vmatrix} A_1 & a_1 & \ldots & a_1 & a_1 \\ a_2 & A_2 & \ldots & a_2 & a_2 \\ \ldots & \ldots & \ddots & \ldots & \\ a_{n-1} & a_{n-1} & \ldots & A_{n-1} & a_{n-1} \\ 1 & 1 & \ldots & 1 & 1 \end{vmatrix}$$

with

$$A_i = r_i \mu - r_i \sum_{k=1}^{n} \gamma_k - \gamma_i \quad \text{for} \quad i = 1, \ldots, n-1$$

and

$$a_i = -\gamma_i.$$

With standard calculations on A it can be easily shown that

$$\det A = \prod_{k=1}^{n-1} (A_k - a_k),$$

so

$$\det \frac{\vartheta \mathbf{P}}{\vartheta \gamma} = 0$$

$$\Leftrightarrow \prod_{k=1}^{n-1} \left(\left(r_k \mu - r_k \sum_{i=1}^{n} \gamma_i \right) - \gamma_k - (-\gamma_k) \right) \left(\sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} r_k (r_i + 1) \gamma_i - \prod_{k=1}^{n} r_k \mu \right) = 0$$

$$\Leftrightarrow \left(\prod_{k=1}^{n} r_k (\mu - \sum_{l=1}^{n} \gamma_l) \right)^{n-1} \left(\sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} r_k (r_i + 1) \gamma_i - \prod_{k=1}^{n} r_k \mu \right) = 0.$$

•

We are now ready to prove the main result of our paper.

Theorem 1.

A necessary and sufficient condition for a set γ of throughputs to be Pareto optimal is

$$\sum_{i=1}^{n} \prod_{\substack{k=1 \\ k \neq i}}^{n} r_k (r_i + 1) \gamma_i = \prod_{k=1}^{n} r_k \mu \tag{5}$$

Proof :

a) Necessary

It is enough to show that the set γ defined by (5) is the only set that achieves $\det \frac{\vartheta \mathbf{P}}{\vartheta \gamma} = 0$[5] which is a direct consequence of the previous lemma and the fact that we are searching for an admissible set of throughputs.

b) Sufficient

Note that for every $i, j = 1, 2, \ldots, n$

$$\frac{\vartheta P_i}{\vartheta \gamma_j} = -\gamma_i^{r_i} < 0, \qquad \text{for } i \neq j$$

and

$$\begin{aligned} \frac{\vartheta P_i}{\vartheta \gamma_i} &= \gamma_i^{r_i - 1} \left(r_i \mu - r_i \sum_{k=1}^{n} \gamma_k - \gamma_i \right) \\ &\overset{(5)}{=} \gamma_i^{r_i - 1} \left(r_i \frac{\sum_{l=1}^{n} \prod_{\substack{k=1 \\ k \neq l}}^{n} r_k \gamma_l}{\prod_{k=1}^{n} r_k} - \gamma_i \right) \\ &= \gamma_i^{r_i - 1} \frac{\sum_{\substack{l=1 \\ l \neq i}}^{n} \prod_{\substack{k=1 \\ k \neq l}}^{n} r_k \gamma_l}{\prod_{\substack{k=1 \\ k \neq i}}^{n} r_k} > 0. \end{aligned}$$

So no change from the points γ given by (5) is going to increase the power of any class, without decreasing the power of some other, i.e. the set γ is a Pareto optimal set.

•

In figures 2a and 2b the set of Pareto optimal throughputs are shown for cases of 2 and 3 classes. An obvious generalization to the n classes case can be done.

The theorem identifies the points where each user cooperates with the other users so as to achieve a performance with the property that if any other solution is used at least one of the users is penalized, in the sense that its performance is deteriorated. Therefore there is no notion of a global or a system optimum. In [4] *Bharath-Kumar* and *Jaffe* give several different definitions for a "system " power. The Pareto optimal points given in theorem 1 have the additional property to maximize this system power in some special cases.

The following corollary provides the results for the case when all users have the same weighting factor on throughput

Corollary 1.

If $r_1 = r_2 = \ldots = r_n = r$ *, then* $\gamma^* = (\gamma_1^*, \gamma_2^*, \ldots, \gamma_n^*)$ *is a Pareto optimal solution iff*

$$\sum_{i=1}^{n} \gamma_i^* = \frac{r\mu}{r+1}. \tag{6}$$

●

If the system power is given by definition 2, where γ is the total throughput and E_T the average system time delay, then the Pareto optimal point given by equation (6) maximizes the system power.

In [7] a blocking procedure to achieve a system optimal point was presented. Packets from all classes are blocked probabilistically independently of their class and are then queued to be served. This probabilistic blocking gives a set of admissible throughputs, which is both a Pareto optimal solution and a system optimum point. The throughputs achieved by this kind of blocking are proportional to the arrival rates of each class of users.

If we define the total power of the system as the sum of the powers of each class then the following proposition holds.

Corollary 2.

Let $r_1 = r_2 = \ldots = r_n = 1$*, and* $P_{\text{sys}} = P_1 + P_2 + \ldots + P_n$ *, then all the Pareto optimal points maximize the system power.*

Proof :

Note that for $r_1 = r_2 = \ldots = r_n = 1$ (6) gives

$$\sum_{i=1}^{n} \gamma_i^* = \frac{\mu}{2} \tag{7}$$

and

$$P_{\text{sys}} = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} \gamma_i(\mu - \sum_{k=1}^{n} \gamma_k) = (\mu - \sum_{i=1}^{n} \gamma_i) \sum_{i=1}^{n} \gamma_i,$$

which is maximized at the Pareto optimal point given by (7)

●

In general it is easy to see that the following corollary also holds.

Corollary 3.

Let the system power be defined as the weighted sum of the powers of each user, i.e.

$$\text{Power}_{\text{system}} = \sum_{i=1}^{n} w_i P_i,$$

where the w_i's are positive constants, that give the weight of each power. Then if $\mathbf{P}^* = (P_1^*, P_2^*, \ldots, P_n^*)$ maximizes the total system power, $\mathbf{P}^*$ is a Pareto optimal set of powers.

From equation (5) as well as from figures 2a and 2b, it can be seen that at certain Pareto optimal points the throughput and/or the power of some of the classes can be very low, even equal to zero at the extreme points of the set. This leads us to the conclusion that the Pareto optimality of a solution may not be able to give a satisfactory solution, if considered alone. In certain applications we may need to put more constraints on the set of optimal solutions than the ones specified by the Pareto optimality. The issue of "fairness" between the several classes is possibly one of these additional requirements. We may not want a certain class to achieve a power or a throughput that is orders of magnitude greater than that of the other classes. In this case the set of admissible points will be a subset of the set of Pareto optimal points. Another restriction could come from the fact that each class might have a desirable power to achieve, in which case we should operate at the Pareto optimal point that minimizes the deviations from the desired operating point.

IV. CONCLUSIONS

Necessary and sufficient conditions have been found for the Pareto optimal points in a computer communications environment, where the power is used as a cost criterion and several classes of users with different performance criteria compete for the resources of a single queue. It has also been proved that in the case of equal weighting factors the system optimization criterion presented in [7] leads to a Pareto optimal point and that when all the weighting factors equal 1 all Pareto optimal points maximize the total power as well.

In [6] an algorithm to achieve a Pareto optimal point has been presented. A search for more efficient algorithms to reach these optimal points seems to be the next step in our research together with the search for Pareto solutions in the more general environment of a Markovian queueing network.

ACKNOWLEDGEMENT

This work was partially supported by the Center for Telecommunications Research at Columbia University under the National Science Foundation Grant # CDR-44-21402.

REFERENCES

[1] Giessler, A., Hañle J., König A. and Pade E., "Free Buffer Allocation - An Investigation by Simulation", *Computer Networks,* Vol. 1, No. 3, July 1978, pp. 199-204.

[2] Kleinrock, L., "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications", *Conference Record of the International Conference on Communications,* pp. 43.1.1-43.1.10, June 1979.

[3] Schmitendorf, W.E., "Cooperative games and vector-valued criteria problems", *IEEE Transactions on Automatic Control,* Vol. AC-18, April 1973.

[4] Bharath-Kumar K. and Jaffe J.M., "A new approach to performance oriented flow control", *IEEE Transactions on Communications,* Vol COM-29, April 1981, pp. 427-435.

[5] Chu, C.K., "On the noninferior set for the systems with vector valued objective functions", *IEEE Transactions on Automatic Control,* Vol. AC-15, October 1970, pp. 591-593.

[6] Cansever, D., "Decentralized algorithms for flow control in networks ", Proceedings of the 25th *Conference on Decision and Control,* Athens, Greece, December 1986.

[7] Douligeris C. and Mazumdar R., "An approach to flow control in an integrated environment," *Submitted for publication.*

[8] Luce R.D. and Raiffa H., *Games and Decisions,* John Wiley & Sons, 1957.

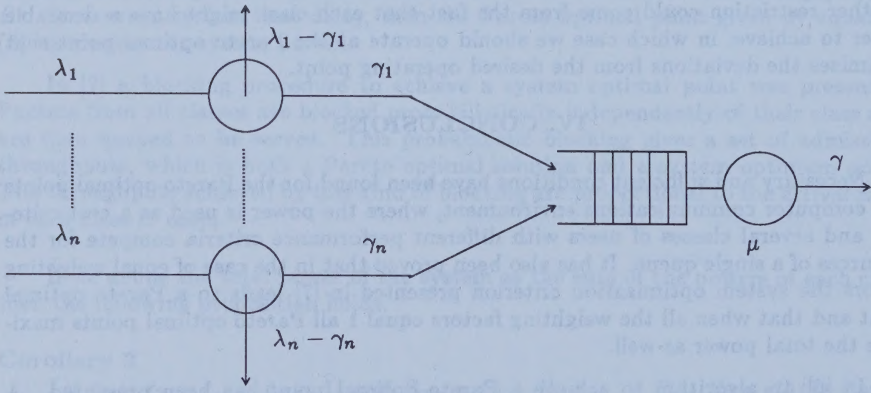


Figure 1. The model

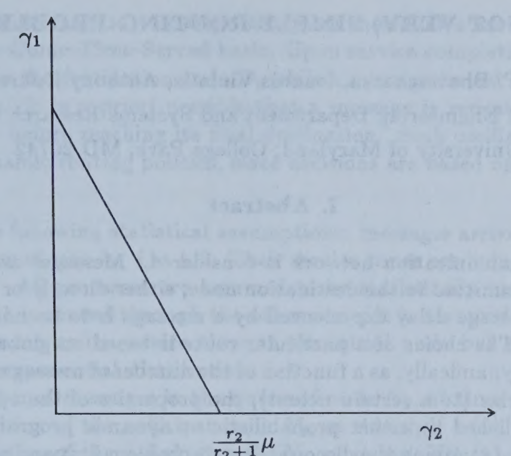

γ_1
γ_2
$\frac{r_2}{r_2+1}\mu$

Figure 2a. The Pareto optimal set for the 2 users case

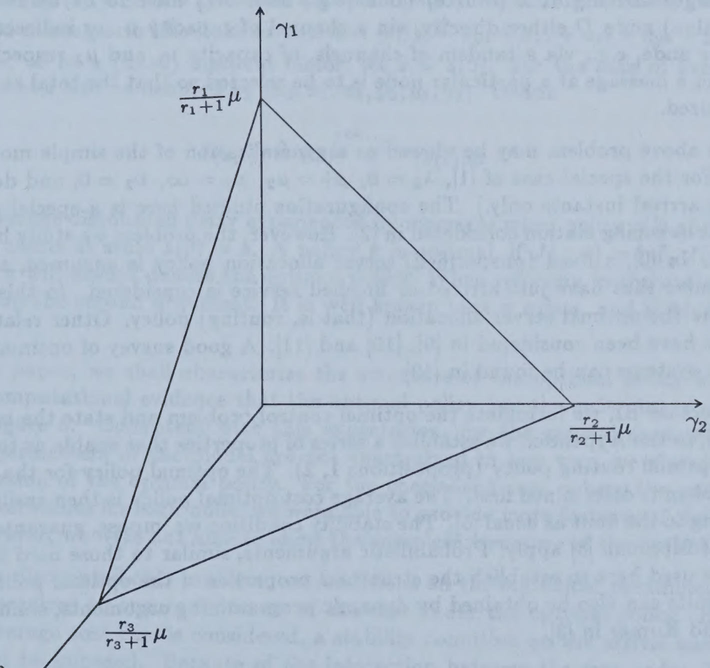

γ_1
$\frac{r_1}{r_1+1}\mu$
$\frac{r_2}{r_2+1}\mu$
γ_2
$\frac{r_3}{r_3+1}\mu$
γ_3

Figure 2b. The Pareto optimal set for the 3 users case

A (NOT VERY) SIMPLE ROUTING PROBLEM

Partha P. Bhattacharya, Ioannis Viniotis, Anthony Ephremides
Electrical Engineering Department and Systems Research Center
University of Maryland, College Park, MD 20742

I. Abstract

A two-node communication network is considered. Messages arriving at a source node have to be transmitted to the destination node, either directly or through the other source node. The average delay experienced by a message is to be minimized by proper selection of routes. The choice of a particular route is based on global information and decisions are taken dynamically, as a function of the number of messages at the two source nodes. We characterize (to a certain extent), the properties of the optimal policy. The results can be established by either probabilistic or dynamic programming arguments. The usual approach of studying the discounted cost problem first and passing to the limit for the average cost case is followed.

II. Introduction

We consider a routing problem in the two-node communication network of figure 1. Messages arriving at a (source) node (e.g., node N_1) have to be transmitted to a (destination) node D either directly, via a channel of capacity μ_1 or indirectly through the other node, e.g., via a tandem of channels, of capacity ν_1 and μ_2 respectively. The routing of a message at a particular node is to be selected so that the total average delay is minimized.

The above problem may be viewed as a generalization of the simple model studied in [1]. (For the special case of [1], $\lambda_2 = 0$, $\mu_1 = \mu_2$, $\nu_1 = \infty$, $\nu_2 = 0$, and decisions are taken at arrival instants only.) The configuration studied here is a special case of the two-node queueing station considered in [2]. However, the problem we study here is quite different. In [2], a fixed (unspecified) server allocation policy is *assumed*, and routing of customers that have just arrived or finished service is considered. In this paper, we *determine* the optimal server allocation (that is, routing) policy. Other related routing problems have been considered in [9], [10] and [11]. A good survey of optimal control of queueing systems can be found in [12].

In section III, we formulate the optimal control problem and state the main results and then, in the Appendix, we establish a series of properties that enable us to characterize the optimal routing policy (propositions 1, 2). The optimal policy for the discounted cost problem is determined first. The average cost optimal policy is then easily obtained, by passing to the limit as usual [5]. The stability condition we impose, guarantees that the results of Lippman [5] apply. Probabilistic arguments, similar to those used by Walrand in [4] are used here to establish the structural properties of the optimal policy. Most of these results can also be obtained by dynamic programming arguments, similar to those of Lin and Kumar in [3].

III. Model Formulation and Structure of the Optimal Policy

The queueing model of the above network is shown in figure 2. Messages arriving at a node immediately join an infinite capacity queue. Each of the queues is served by two links, of capacities μ_1, μ_2, ν_1, ν_2 respectively, as shown. We assume that the ν–links operate on a Last–Come–First–Served basis. Upon service completion, a message served by a μ–link reaches the destination node D, while a message served by a ν–link is queued at the other node. It is a priori possible that a message is repeatedly routed between nodes N_1 and N_2 before reaching its final destination. Such oscillatory behavior is not uncommon in dynamic routing policies, since decisions are based on current information [6].

We adopt the following statistical assumptions: messages arrive at node i according to a Poisson stream of rate λ_i, $i = 1, 2$. Their service times are independent, exponential random variables. All processes are assumed to be statistically independent from each other. If a message is routed through the other node, it is assumed that its service time there is not the same as its service time at the initial node but rather independently selected from the exponential distribution. Let

x_{0t} : number of messages in the queue of node N_1, at time t.
x_{1t} : condition of the μ–link at node N_1, at time t; if 1 (0), link is busy (idle).
x_{2t} : condition of the ν–link at node N_1, at time t; if 1 (0), link is busy (idle).

Define y_{0t}, y_{1t}, y_{2t} accordingly, for node N_2. Under the statistical assumptions made above, $z(t) \triangleq (x_{0t}, x_{1t}, x_{2t}, y_{0t}, y_{1t}, y_{2t})$ is a suitable state description of the system. Clearly, the state space is $Z = \{0, 1, 2, \cdots\} \times \{0, 1\}^2 \times \{0, 1, 2, \cdots\} \times \{0, 1\}^2$.

A (nonanticipative) policy γ is any rule which, at $t \geq 0$, decides (on the basis of the trajectory $\{z(s), s \leq t\}$ it generates and the "history" of actions it takes) whether to send queued messages to idle links or not. Let $|z(t)| \triangleq x_{0t} + x_{1t} + x_{2t} + y_{0t} + y_{1t} + y_{2t}$; let $0 < \alpha < \infty$ be a (fixed) discount factor; let $z \triangleq z(0)$. For the sake of brevity, the components of z will be denoted by $(x_0, x_1, x_2, y_0, y_1, y_2)$. Define

$$J(z, \gamma) \triangleq E \int_0^\infty e^{-\alpha t} |z(t)| dt \tag{1}$$

to be the average discounted delay experienced by messages when policy γ is applied and the system starts at state $z(0) = z$. A policy π is optimal, if $J(z, \pi) = \inf_\gamma J(z, \gamma)$. In the sequel, π will always denote the optimal policy. When referring to the optimal cost, we shall drop the second argument. It is well known that π exists, and is Markov and stationary.

In this paper, we shall characterize the structure of the optimal policy π. There is strong computational evidence that the optimal policy has the switching–curve form shown in figure 5. Consistent with this expectation, for the general case, where the statistical parameters of the model are not constrained in any way, we give a limited characterization of the optimal policy. For the specialized case, where the parameters have identical values for each node, we were able to provide more features of the optimal policy. However, we were not able to show the complete structure of the optimal policy.

For the discounted cost problem, no conditions on the statistical parameters of the model are required to ensure finiteness of the cost under the optimal policy. However, when the average cost case is considered, a stability condition on the arrival and service rates should be imposed. Because of the interaction between the two nodes, it is not clear which condition is the "best", i.e., what are the maximum arrival rates λ_1, λ_2 that guarantee stability for given link capacities. Obviously, the "intuitive" condition $\lambda_1 + \lambda_2 < \mu_1 + \mu_2$ is not sufficient to guarantee stability of both queues. On the other hand, $\lambda_i < \mu_i$, $i = 1, 2$, is a sufficient condition, even though it is "too restrictive".

Furthermore, $\lambda_1+\nu_2 < \mu_1$, $\lambda_2+\nu_1 < \mu_2$ is another condition sufficient for stability, even though, it too, for some values of the parameters might become even more restrictive than the first one.

In the Appendix, we determine several properties of the optimal policy and the cost function (lemmas 1–7). The structural form of π can then be determined in part.

Let $(\pi(x_0), \pi(x_1), \pi(x_2), \pi(y_0), \pi(y_1), \pi(y_2))$ denote the state that results when π is applied at state $z = (x_0, x_1, x_2, y_0, y_1, y_2)$. Define the following subsets of the state space Z:

$$S_0 \triangleq \{z \in Z \mid x_2 = y_2 = 0, \pi(x_2) = \pi(y_2) = 0\}$$

$$S_1 \triangleq \{z \in Z \mid x_2 = 0, \pi(x_2) = 1, \pi(y_2) = y_2\}$$

$$S_2 \triangleq \{z \in Z \mid y_2 = 0, \pi(x_2) = x_2, \pi(y_2) = 1\}$$

$$S_3 \triangleq \{z \in Z \mid x_2 = y_2 = 0, \pi(x_2) = \pi(y_2) = 1\}$$

$$Z_1 \triangleq \{z \in Z \mid x_1 = y_1 = 0, \pi(x_1) = \pi(y_1) = 0, \ or \ \pi(x_1) \neq \pi(y_1)\}$$

$$Z_2 \triangleq \{z \in Z \mid x_1 = y_1 = 0, \pi(x_1) = \pi(y_1) = 1\}$$

In other words,

S_0 is the set of states at which the optimal policy idles both ν–links,
S_1 (S_2) is the set of states at which the optimal policy activates the ν–link at node N_1 (N_2) only, even if the ν–link at the other node is idle,
S_3 is the set of states at which the optimal policy activates both ν–links,
Z_1 is the set of states at which the optimal policy does not activate both μ–links at the same time, even though they are *both* idle at a decision instant, and finally,
Z_2 is the set of states at which the optimal policy activates both μ–links.

Notice that these subsets need not be necessarily disjoint.

The following propositions are first shown that summarize the properties exhibited in detail in the Appendix. Proposition 1 refers to the general case; proposition 2 holds true only for the special "symmetric" model for which $\lambda_1 = \lambda_2 = \lambda$, $\mu_1 = \mu_2 = \mu$, $\nu_1 = \nu_2 = \nu$ and $\mu > \nu$. Figures 3 and 4 depict the properties stated in propositions 1 and 2 respectively.

Proposition 1: a) $Z_2 \neq \emptyset$, (i.e., there are states at which both μ–links should be activated.)

b) If $\mu_i > \nu_i$ then $Z_1 = \emptyset$, (i.e., the μ–links should be *always* busy.)

c) $S_1 \neq \emptyset$; $S_2 \neq \emptyset$; $S_3 = \emptyset$, (i.e., there exist states at which a ν–link should be activated, while at no state *both* ν–links should be *simultaneously* activated.)

Proof: Part (a) follows from lemma 1. Part (b) follows from lemma 2. That S_3 is empty follows from lemma 3, while the rest of part (c) follows from lemma 4.

Remarks: 1) The more interesting interpretation of the properties stated in proposition 1 can be given when the profile of the state space on the plane (x_0, y_0) of queue sizes is considered, parametrized by the condition (x_1, x_2, y_1, y_2) of the four servers. In this plane, there exists a curve C, in general a function of x_1, x_2, y_1, y_2, with a form as depicted in figure 3, such that for all queue sizes x_0, y_0 "above" the curve, the state $(x_0, x_1, x_2, y_0, y_1, y_2)$ belongs to the set Z_2.

2) Even though it is intuitively obvious that idling *both* μ–links should not be optimal, we have not been able to show that there exists no state in Z_1 with this property. Another rather intuitive fact that we have not been able to show is that the fastest μ–link should be busy whenever possible.

In the special symmetric case, the optimal policy has the properties described in proposition 2 as well.

Proposition 2: If $\lambda_1 = \lambda_2 = \lambda$, $\mu_1 = \mu_2 = \mu$, $\nu_1 = \nu_2 = \nu$, and $\mu > \nu$, then

a) $S_0 \neq \emptyset$, (i.e., there are states at which it is optimal to keep *both* ν–links idle.)

b) When it is optimal to activate the ν–link at node N_1 (N_2), the queue size at that node should be larger than the queue size at node N_2 (N_1), i.e., $z \in S_1 \Rightarrow x_0 > y_0$.

c) If $z = (x_0, x_1, 0, y_0, y_1, y_2) \in S_1$ then $\bar{z} = (y_0, y_1, y_2, x_0, x_1, 0) \in S_2$.

Proof: a) That S_0 is nonempty follows from lemma 7. Assertion (b) follows from lemma 7b. Assertion (c) follows from lemma 5. △

Remark: An interpretation of these properties in the (x_0, y_0) plane (see figure 4), is the following: when $x_0 \approx y_0$, (e.g., when $x_0 = y_0$, or $x_0 = y_0 \pm 1$), the ν–links should not be activated. Part (b) states that, since $\mu_1 = \mu_2$, the ν_1–link should never be activated whenever node N_2 is more loaded than node N_1. Recall that since $\mu > \nu$, the direct links should be busy whenever the queues are nonempty (part (b) of proposition 1).

That the optimal policy for the average cost case has the same properties as well, is an immediate consequence of corollary 1 in [5], since the stability condition guarantees that a policy which incurs finite average delays exists.

IV. Conclusions

We have considered an optimal routing problem in a two–node communication network. Decisions were dynamic in nature and based on global state information. We have exhibited part of the structure of the optimal policy for the general and a specialized case, as described in figures 3 and 4. Probabilistic arguments have been presented to establish the properties of the optimal policy. Dynamic programming arguments can be used equally well to establish all the results, except lemmas 1 and 4. We still conjecture that the unproven properties of the optimal policy are correct, namely that it has a "switching–curve" form as shown in figure 5.

Appendix

In this appendix we are going to present the proofs of the properties of the optimal policy stated in propositions 1 and 2. We do so in a sequence of seven lemmas.

The first four lemmas hold true for the general case; lemmas 5 and 6 hold true only for the special case $\lambda_1 = \lambda_2 = \lambda$, $\mu_1 = \mu_2 = \mu$, $\nu_1 = \nu_2 = \nu$, while for lemma 7 we further require that $\mu > \nu$.

Since there are four servers in the system, a total of 16 operators that describe assignment of customers to servers can be defined, as in [3]. For the sake of brevity, we define here only those that will be explicitly used in the presentation, namely:

$$P_0(x_0, x_1, x_2, y_0, y_1, y_2) = (x_0, x_1, x_2, y_0, y_1, y_2)$$

$P_4(x_0, x_1, 0, y_0, y_1, y_2) = (x_0 - 1, x_1, 1, y_0, y_1, y_2)$
$P_5(x_0, x_1, 0, y_0, y_1, 0) = (x_0 - 1, x_1, 1, y_0 - 1, y_1, 1)$
that is, P_0 represents "idling", P_4 represents forwarding a customer from the x–side to ν–link and P_5 represents forwarding one customer from each node to its ν–link.

The following lemmas characterize some interesting properties of the optimal cost function $J(z)$ and the optimal policy π. Properties are stated for the x–side queue (node N_1) only. Similar statements hold true for the y–side (node N_2) as well. Proofs which are easy or similar to previous ones are omitted.

The first lemma states the fact that the direct route at a node should be activated when the load at this node becomes too large. Note that since $\mu_1 \neq \mu_2$ in general, it is not clear that a direct route should be activated when the queue size is nonzero. Routing a message through the other node could (at least in principle) decrease the total waiting time.

Lemma 1: For any (x_2, y_0, y_1, y_2), there exists an integer $m = m(x_2, y_0, y_1, y_2) < \infty$ such that π activates the μ_1–link whenever the queue size at node N_1 is at least m.

Proof: Suppose in the contrary that, for the given x_2, y_0, y_1, y_2, m is not finite. Let $A \overset{\Delta}{=} \{z \in Z \mid x_1 = 0, \pi(x_1) = 0\}$. Then for all x, by assumption, $(x, 0, x_2, y_0, y_1, y_2) \notin A$.

Let $\tilde{\pi}$ be a policy which at $t = 0$ activates the μ_1–link at node N_1, and follows π otherwise. Let C denote the message forwarded for service to the μ_1–link at $t = 0$ (under $\tilde{\pi}$). For convenience, assume that C is served under π only when no other messages wait in the queue. Define the following times:

$\tilde{X}_{\mu_1}$: C's departure instant, under $\tilde{\pi}$.

τ: first time a state in the set A is visited under π.

T: C's departure instant under π.

σ: first time that π forwards C for service (to any link).

Clearly, in $(0, \sigma)$, $\tilde{\pi}$ can follow π's actions. At $t = \sigma$, $\tilde{\pi}$ has to idle. For fixed x, let $G(x)$ be the net gain of $\tilde{\pi}$ over π. Obviously, for all realizations for which $\sigma \geq \tau$, $\tilde{\pi}$ cannot lose to π. Consider the realizations for which $\sigma < \tau$. $\tilde{\pi}$ gains if $\tilde{X}_{\mu_1} < T$ and loses otherwise. Therefore,

$$G(x) \overset{\Delta}{=} E\tilde{G}(x) = E\{1(\sigma < \tau)\tilde{G}(x)\} + E\{1(\sigma \geq \tau)\tilde{G}(x)\} \tag{1}$$

We can easily show that there exists $c > 0$, independent of x, such that

$$E\{1(\sigma \geq \tau)\tilde{G}(x)\} \geq E\{1(\sigma \geq \tau)c\} \tag{2}$$

Also it is clear that

$$E\{1(\sigma < \tau)\tilde{G}(x)\} \geq E\{1(\sigma < \tau)\{1(\tilde{X}_{\mu_1} < T)\int_{\tilde{X}_{\mu_1}}^{T} e^{-\alpha t}dt - 1(\tilde{X}_{\mu_1} \geq T)\int_0^{\tilde{X}_{\mu_1}} e^{-\alpha t}dt\}\}$$

$$\overset{\Delta}{=} E\{1(\sigma < \tau)G_1(T)\}$$

For the realizations that $\sigma < \tau$, all x messages initially present at the queue of node N_1 have to be served by the ν_1–link. Therefore, $T \geq \sum_1^x Y_i \overset{\Delta}{=} T'$, where Y_i's are exponentially distributed with parameter ν_1. Since $G_1(T)$ decreases as T decreases, it is clear that

$$E\{1(\sigma < \tau)\tilde{G}(x)\} \geq E\{1(\sigma < \tau)G_1(T')\} \tag{3}$$

Since T' increases to ∞, as x increases, and $\tilde{X}_{\mu_1} < \infty$ a.s., one easily concludes from (2) and (3) that there exists a finite m, (in general a function of x_2, y_0, y_1, y_2), such that for all $x \geq m$, $G(x) > 0$, q.e.d.

An interesting special case is the one for which $\mu_i > \nu_i$, $i = 1, 2$, i.e., the capacity of the direct route at each node is greater than the capacity of the indirect one. In this case, a stronger property than that described in lemma 1 can be shown, namely that the direct routes should kept busy whenever possible.

Lemma 2: For $\mu_i > \nu_i$, $i = 1, 2$ and all $x_0 \geq 1, y_0, y_1, y_2$, the following properties are true:

a) $\pi(x_0, 0, 0, y_0, y_1, y_2) \neq (x_0 - 1, 0, 1, \bar{y}_0, \bar{y}_1, \bar{y}_2)$.
b) $\pi(x_0, 0, 1, y_0, y_1, y_2) \neq (x_0, 0, 1, \bar{y}_0, \bar{y}_1, \bar{y}_2)$.
c) $\pi(x_0, 0, 0, y_0, y_1, y_2) \neq (x_0, 0, 0, \bar{y}_0, \bar{y}_1, \bar{y}_2)$.

Proof: Only the proof of part (a) will be presented here, since the other proofs are similar.

Suppose $\pi(x_0, 0, 0, y_0, y_1, y_2) = (x_0 - 1, 0, 1, \bar{y}_0, \bar{y}_1, \bar{y}_2)$. Let $\tilde{\pi}$ at $t = 0$ activate the μ-link at node N_1. Let T_{μ_1} (T_{ν_1}) be the first μ-link (ν-link) completion instant under $\tilde{\pi}$ (π) at node N_1. By stochastic domination, we may assume that $T_{\mu_1} < T_{\nu_1}$ for all realizations. Let σ be the first time π activates the μ-link at node N_1. It is clear that for the realizations for which $T_{\mu_1} < \sigma$, $\tilde{\pi}$ gains over π. Suppose $\sigma < T_{\mu_1}$. Let $\tilde{\pi}$ activate the ν-link at $t = \sigma$. Then one can easily see that $\tilde{\pi}$ can not lose to π for all such realizations. Finally, if $T_{\mu_1} = \sigma$, one can again check that $\tilde{\pi}$ achieves a strict gain over π.

Therefore, in all cases, $\tilde{\pi}$ strictly improves π, which contradicts the assumed optimality of π.

The next lemma asserts that activating *both* ν-links at the same time cannot be optimal. This does not mean, however, that there exist no states for which the optimal policy activates a ν-link while the other ν-link is active.

Lemma 3: $J(P_5(x_0, x_1, 0, y_0, y_1, 0)) \geq J(P_0(x_0, x_1, 0, y_0, y_1, 0))\ \forall\ x_0 \geq 1,\ y_0 \geq 1.$

Proof: Let $z = (x_0, x_1, 0, y_0, y_1, 0)$. Assume that at $t = 0$, π takes action P_5. Let $\tilde{\pi}$ be another policy which at $t = 0$ takes action P_0. Let τ be the first time a message finishes service at one of the two ν-links (say the link at node N_1), under π. Clearly $\tilde{\pi}$ can follow π in $(0, \tau)$. At $t = \tau$, the states under π and $\tilde{\pi}$ are $(x - 1, \cdot, 0, y, \cdot, 1)$ and $(x, \cdot, 0, y, \cdot, 0)$ respectively. Let σ be the time at which the message at the ν-link of node N_2 finishes service. Clearly, at $t = \sigma$ the states under both policies are the same, and therefore $J(P_5 z) \geq J(P_0 z)$, q.e.d.

The next lemma asserts that for any configuration at node N_2, when the queue at node N_1 becomes too "congested", the ν-link at node N_2 should be activated, in an effort to "balance" the workload between the two nodes. The proof parallels in most parts that of lemma 1, therefore it will be omitted.

Lemma 4: For any x_1, y_0, y_1, y_2, there exists an integer $m = m(x_1, y_0, y_1, y_2) < \infty$ such that the optimal policy activates the ν_1-link whenever the queue size at node N_1 is at least m.

The next two lemmas hold true for the special symmetric case where the parameters of the statistical model have identical values for each node.

Property (a) of lemma 5 states that the optimal cost function is symmetric while property (b) essentially says that, since the message served by a ν–link will arrive at the other node after some delay, the cost can only decrease, if this message is already queued at the other node (and thus available immediately for possible service).

Lemma 5: The following properties are true:

a) $J(x_0, x_1, x_2, y_0, y_1, y_2) = J(y_0, y_1, y_2, x_0, x_1, x_2)$.

b) $J(x_0, x_1, 1, y_0, y_1, y_2) \geq J(x_0, x_1, 0, y_0 + 1, y_1, y_2)$, with equality if $\nu > \mu$.

The next lemma states the intuitive fact that the cost incurred when starting from an initial state z decreases as z becomes more "balanced".

Lemma 6: $J(x-1, x_1, x_2, y+1, y_1, y_2) \leq J(x, x_1, x_2, y, y_1, y_2)$, $1 \leq y < x$.

b) Let $x + y = x' + y'$ and $0 \leq x - y < x' - y'$. Then $J(x, 1, x_2, y, 1, y_2) \leq J(x', 1, x_2, y', 1, y_2)$.

Proof: Let $z_0 = (x, x_1, x_2, y, y_1, y_2)$, $z_1 = (x-1, x_1, x_2, y+1, y_1, y_2)$. Apply π when starting from z_0 at $t = 0$, and $\tilde{\pi}$ when starting from z_1. Let $\tilde{\pi}$ keep the μ–links busy and activate a ν–link only when π does so (and of course when there are messages in the queue.) Define

σ_x : first time to hit $(0, 0, \bar{x}_2, \bar{y}_0, \bar{y}_1, \bar{y}_2)$ under π.

σ_y : first time to hit $(\bar{x}_0, \bar{x}_1, \bar{x}_2, 0, 0, \bar{y}_2)$ under $\tilde{\pi}$.

Since $x \geq y$, one can show easily that $\sigma_x \geq_{st} \sigma_y$. Let G be the net gain of $\tilde{\pi}$ over π. We have

$$G \triangleq E\{1(\sigma_y < \sigma_x) \int_{\sigma_y}^{\sigma_x} e^{-\alpha t} dt - 1(\sigma_x < \sigma_y) \int_{\sigma_x}^{\sigma_y} e^{-\alpha t} dt\} = E \int_{\sigma_y}^{\sigma_x} e^{-\alpha t} dt > 0$$

as it is easily seen, since $\sigma_x \geq_{st} \sigma_y$. Since $\tilde{\pi}$ need not be optimal, $J(x-1, x_1, x_2, y+1, y_1, y_2) \leq J(x, x_1, x_2, y, y_1, y_2)$, q.e.d.

Assertion (b) now follows easily, by repeatedly applying the result in part (a). $\triangle$

Recall that for the case $\mu > \nu$, the μ–links should be activated whenever possible. Therefore, we can restrict attention to states z with $x_1 = 1, y_1 = 1$, whenever $x_0 \geq 1, y_0 \geq 1$. Since two of the state components are fixed, the following lemma is best interpreted in the (x_0, y_0) plane. It basically states that a) when the queue sizes are the same, it does not pay to activate a ν–link, b) the set of states at which it is possibly optimal to activate the x–side (y–side) ν–link lies necessarily "below" (above) the $x_0 = y_0$ diagonal, c) the ν–link should not be activated when the queue sizes differ exactly by one.

Lemma 7: If $\mu > \nu$, the following properties are true:

a) $J(P_4(x_0, 1, 0, x_0, 1, y_2)) \geq J(P_0(x_0, 1, 0, x_0, 1, y_2)) \quad \forall\, x_0 \geq 1$

b) $J(P_4(x_0, 1, 0, y_0, 1, y_2)) \geq J(P_0(x_0, 1, 0, y_0, 1, y_2))$ for $1 \leq x_0 \leq y_0$

c) $J(P_4(x_0 + 1, 1, 0, x_0, 1, y_2)) \geq J(P_0(x_0 + 1, 1, 0, x_0, 1, y_2))$

Proof:

a) $J(P_4(x_0,1,0,x_0,1,y_2)) = J(x_0-1,1,1,x_0,1,y_2) \geq J(x_0-1,1,0,x_0+1,1,y_2)$
$\geq J(P_0(x_0,1,0,x_0,1,y_2))$ (the last inequality follows from lemma 6).

b) $J(P_4(x_0,1,0,y_0,1,y_2)) = J(x_0-1,1,1,y_0,1,y_2) \geq J(x_0-1,1,0,y_0+1,1,y_2) \geq$
$J(P_0(x_0,1,0,y_0,1,y_2))$ (the last inequality follows from lemma 6).

c) $J(P_4(x_0+1,1,0,x_0,1,y_2)) = J(x_0,1,1,x_0,1,y_2) \geq J(x_0,1,0,x_0+1,1,y_2)$
$= J(x_0+1,1,0,x_0,1,y_2) = J(P_0(x_0+1,1,0,x_0,1,y_2))$ $\triangle$

These lemmas lead directly to the propositions stated in section III.

REFERENCES

[1] A. Ephremides, P. Varaiya and J. Walrand, "A Simple Dynamic Routing problem," IEEE Transactions on Automatic Control, vol. AC-25, no. 4, pp. 690–693, August 1980.

[2] B. Hajek, "Optimal Control of Two Interacting Service Stations," IEEE Transactions on Automatic Control, vol. AC-29, no. 6, pp. 491–498, June 1984.

[3] W. Lin and P. R. Kumar, "Optimal Control of a Queueing System with Two Heterogeneous Servers", IEEE Transactions on Automatic Control, vol. AC-29, no. 8, pp. 696–703, August 1984.

[4] J. Walrand, "A note on Optimal Control of a Queueing System with Two Heterogeneous Servers", Systems and Control Letters, no 4, pp. 131–134, 1984.

[5] S. A. Lippman, "Semi-Markov Decision Processes with Unbounded Rewards", Management Science, vol. 19, pp. 717–731, 1973.

[6] D. Bertsekas, "Dynamic Behavior of Shortest Path Routing Algorithms in Communication Networks," IEEE Transactions on Automatic Control, vol. AC-27, no. 1, pp. 60–74, January 1982.

[7] D. Stoyan, "Comparison Methods for Queues and Other Stochastic Models," J. Wiley and Sons, 1983.

[8] D. R. Jacobs and S. Schach, "Stochastic Order Relationships Between $GI|G|k$ Systems," Annals of Mathematical Statistics, vol. 43, no 5, pp. 1623–1633, 1972.

[9] R. Weber, "On the Optimal Assignment of Customers to Parallel Servers," Journal of Applied Probability, vol. 15, pp. 406–413, 1978.

[10] E. Davis, "Optimal Control of Arrivals to a Two–server Queueing System with Separate Queues," Ph. D. Dissertation, North Carolina State University, Raleigh, NC 1977.

[11] W. Winston, "Optimality of the Shortest–Processing–Time Discipline," Journal of Applied Probability, vol. 14, pp. 181–189, 1977.

[12] S. Stidham, "Optimal Control of Admission to a Queueing System," IEEE Trans. on Automatic Control, vol. AC–30, no 8, pp. 705–713, 1985.

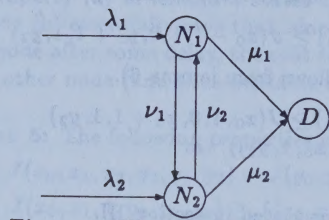

λ_1
N_1
μ_1
ν_1
ν_2
D
λ_2
μ_2
N_2

Figure 1. The network model.

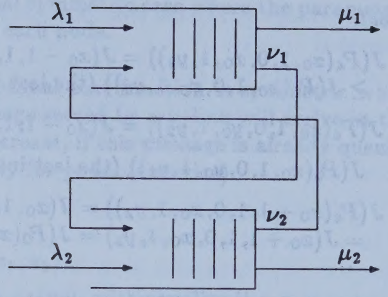

λ_1
μ_1
ν_1
ν_2
λ_2
μ_2

Figure 2. The queueing network.

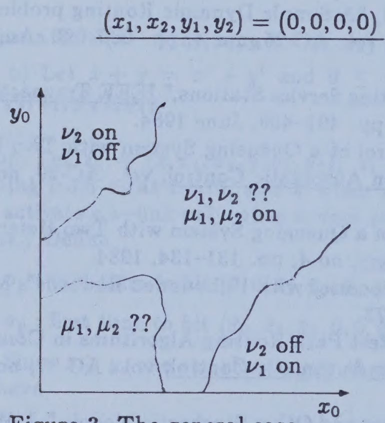

$(x_1, x_2, y_1, y_2) = (0,0,0,0)$
y_0
ν_2 on
ν_1 off
ν_1, ν_2 ??
μ_1, μ_2 on
μ_1, μ_2 ??
ν_2 off
ν_1 on
x_0

Figure 3. The general case.

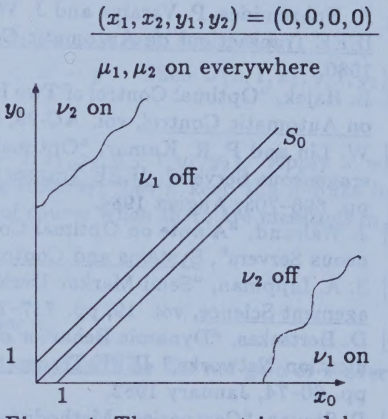

$(x_1, x_2, y_1, y_2) = (0,0,0,0)$
μ_1, μ_2 on everywhere
y_0
ν_2 on
S_0
ν_1 off
ν_2 off
1
ν_1 on
1
x_0

Figure 4. The symmetric model.

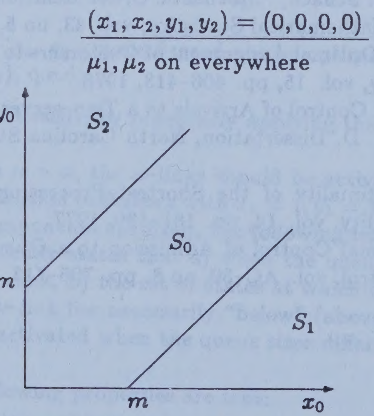

$(x_1, x_2, y_1, y_2) = (0,0,0,0)$
μ_1, μ_2 on everywhere
y_0
S_2
S_0
m
S_1
m
x_0

Figure 5. Conjectured structure of the optimal policy.

TERMINATION DETECTION IN A BROADCAST NETWORK

TEN H. LAI and TAO H. YANG
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

ABSTRACT

This paper studies the problem of deciding whether a process of a distributed system is terminated in a broadcast network environment. We show that, by discriminating the underlying messages between two classes, the complexity of termination detection will most likely be reduced. We then propose three termination detectors, each shown to be message optimal. For a rich class of distributed systems, our algorithms require no overhead messages at all—all control messages can be piggybacked on underlying messages. The algorithms can be used to decide if the entire system is terminated. When so used, the algorithms are also message optimal and more efficient than existing algorithms, which are designed especially for use in a point-to-point network. Both static and dynamic systems are considered in this paper.

1. INTRODUCTION

One of the most fundamental problems in distributed programming is that of ensuring that a system is able to know whether it has terminated. A distributed system with this capacity is said to be *terminable*; otherwise, it is *nonterminable*.

Francez et al. [6] observed that constructing a nonterminable system may be much easier than a terminable one, and thus suggested relieving a programmer from the problem of termination detection by devising a *termination detection algorithm* (or *termination detector*) that, when superimposed on a possibly nonterminable distributed system, automatically makes the system terminable.

A number of termination detectors have been proposed for various models of distributed systems, including:

– [2], [3], [12], [15] that work for distributed systems with special properties such as diffusion;
– [2], [4], [5], [6], [7], [12], [16] that work for systems employing synchronous communications;
– [1], [13] that work for systems with asynchronous first-in-first-out communications;
– [8], [9], [11] that work for systems with non-first-in-first-out communications. [9] considers static systems as well as dynamic systems.

All of these algorithms were designed especially for systems with point-to-point communications. Although they can be easily modified so as to work for systems with broadcast communications, algorithms so obtained tend to be inefficient as they do not utilize the "broadcast" power offered by a broadcast network. Indeed, these algorithms (except for those that work only for special systems), when applied to a distributed system employing broadcast communication, require $O(mn)$ or even $O(mn^2)$ overhead messages, where n and m are, respectively, the number of processes and the number of messages received in the underlying system.

Since a great number of existing distributed systems (e.g., those running on a local area network) are based on broadcasting communication media such as contention bus, token ring, and token-passing bus [14], it is of practical as well as theoretical interest to devise efficient termination detectors exclusively for such systems.

In this paper, we develop three such termination detectors, two for static systems and the other for dynamic systems. Each algorithm is shown to be optimal, measured by the number of overhead messages introduced by the algorithm. More specifically, we show that any termination detection algorithm in worst cases requires l overhead messages, where l is the total number of times for which the processes change state from active to idle or dead; and in the worst cases, our algorithms require no more than l overhead messages.

We also propose a simple strategy that may help reducing the message complexity of distributed termination. The problem of distributed termination detection normally requires each individual process of a distributed system to detect the termination of the system. In most (if not all) applications, however, it is not necessary for a process to know if the system has terminated; instead, it suffices for each process to know whether the process itself is terminated. We show that, by distinguishing the underlying messages between two classes—α-messages and β-messages, the weaker version of termination detection is computationally easier than the original version. The fewer α-messages are used, the easier is termination detection (the weaker version). We shall point out what messages may be regarded as β-messages, and what may not.

The rest of this paper is organized as follows. Section 2 defines the problem of termination detection. Section 3 proposes an approach to reducing the complexity of termination detection. Section 4 presents basic concepts of the algorithms. Sections 5 - 7 describe three termination detectors, which are shown to be message optimal in Section 8. All proofs of lemmas, propositions and theorems are omitted in this paper, but can be found in [10].

2. DEFINITIONS AND ASSUMPTIONS

A *distributed system* consists of a collection of processes which communicate by messages. We assume that the processes are at a higher layer (e.g., the application layer) in the ISO network architecture, and the underlying communication medium is a broadcast channel such as a contention tree/bus or a token ring [14]. The communication medium at the physical layer is subject to contention and errors, but messages are delivered between processes (at the application layer) without errors, without duplicates, and in order. The delay time experienced by a message is arbitrary but finite.

A message may have multiple destinations. It is assumed in the physical layer that each time a message is transmitted over the channel it is received correctly by either all or none of its destination nodes; that is, if it is received correctly by one then it is received correctly by all. (This assumption is realistic in a token ring or a contention bus [14].)

Consider message m_p from process p to process r and message m_q from process q to process r, where $p \neq q$. We assume that if m_p arrives, without errors, at the physical layer of r earlier than m_q, then m_p is delivered to r before m_q.

The following lemma follows directly from the above assumptions.

Lemma 0. *Let p, q, r be processes. Suppose that p sends messages m_1 and m_2 in that order with m_1 addressed to q and m_2 to both q and r, and suppose r, after receiving m_2, sends message m_3 to q. Then q receives m_1 before it receives m_3.*

Note that if m_2 is addressed to r only then the lemma is no longer true.

Each message is associated with an atomic action, which is executed without interruption immediately after the message is received.

During the operation of a distributed system, each process is either *active* or *idle*, with all processes initially being active. An active process is free at any moment to send a message, to receive (and execute) an arriving message, or to change state to idle. All these actions are atomic and cannot be interrupted. Thus, when a process is executing a message, unless required by the message, the process cannot, for example, send or receive another message. An idle process is free at any moment to receive an arriving message, but it cannot send a message. An idle process becomes active on receiving a message; and after the actions of all received messages have been executed, the process is free to become idle.

A distributed system is said to be *terminated* iff all processes are idle and no messages are still in transit in the system.

Throughout this paper, we assume that every distributed system eventually becomes terminated. Under this assumption, a process of a distributed system is said to be *terminated* iff it is at present idle and will stay idle for the rest of the system's lifetime. That is, a process terminates when it becomes idle for the last time. Note that a process may terminates before the system does. Also note that a system is terminated iff every individual process in the system is terminated.

The problem of *system termination detection* is to devise an algorithm that, once the system terminates, is able to detect the occurrence of termination within a finite amount of time. The problem of *process termination detection* is to devise an algorithm that, once an individual process terminates, can detect the process's termination within a finite amount of time.

Thus, given a distributed system P, we are interested in devising another distributed system $\overline{P} = \{\overline{p} : p \in P\}$ so that $\overline{p}$ can determine if process p (or system P) is terminated.

Conceptually, p and $\overline{p}$ are two distinct processes running concurrently at the same node and communicating with each other through shared variables. In practice, it is usually more efficient to implement p and $\overline{p}$ together as a single process, with messages of p and messages of $\overline{p}$ sharing the same incoming (outgoing) queue.

Messages exchanged between processes of P are called *underlying messages.* Messages exchanged between processes of $\overline{P}$ are called *control messages.* In our algorithms, a control message may *piggyback* on an underlying message (i.e., be attached to an underlying message and broadcast as a part of the latter). A control message that does not piggyback on an underlying message is said to be an *overhead message.*

We assume that the overhead of piggybacking a message is negligible if the piggybacked message is short. In this case, the performance of a termination detector can be measured by the total number of overhead messages used by the detector.

3. AN APPROACH TO REDUCING THE PROBLEM'S COMPLEXITY

As mentioned in the introduction, many applications require only process (rather than system) termination detection. For instance, if the purpose of termination detection is for release of resources held by a process, then process termination detection is sufficient.

The problem of process termination detection is in general not easier than the problem of system termination detection, for a terminated process has no knowledge of its own termination until the time it knows the system is terminated. However, as to be seen in what follows, if users of a termination detector can provide such information as whether the execution of a received message requires send of another message, process termination detection may require fewer control messages than system termination detection.

We first define a *refined* distributed system. (For discrimination, a distributed system as defined in the previous section will be referred to as a *primitive* system.) In a refined distributed system, each process is either *active*, *passive*, or *idle*, and the processes communicate via two types of messages: α-messages and β-messages.

An active process is free to send a message, to receive an arriving message, or to become idle; but it cannot change state to passive. A passive or idle process is free to receive an arriving message, but neither can send a message. On receiving an α-message, a passive or idle process becomes active, and after the actions of all received messages have been executed the process is free to become idle. On receiving a β-message, an idle process becomes passive. A passive process is free to become idle after all received messages have been executed.

Like primitive systems, a refined distributed system is *terminated* iff every process is idle and no messages are in transit; and a process is terminated iff it is idle and will stay idle thereafter.

Let P be a primitive distributed system. Let B be the set of all messages *msg* sent in P such that no message needs to be sent in execution of *msg*. Suppose that a subset $B' \subseteq B$ is given together with P. Then P can be transformed (using a simple translator) into a refined system P' as below:

1. Call messages in B' β-messages and those not in B' α-messages.
2. For each process $p \in P$, introduce a new variable $state'(p)$. Initially, $state'(p) =$ active.
3. Whenever p becomes idle, set $state'(p)$ to idle.
4. Whenever p receives an α-message, set $state'(p)$ to active.
5. Whenever p becomes active on receipt of a β-message, set $state'(p)$ to passive.

Let P' be obtained as above, with $state'(p)$ regarded as the state of p for every process p. Then P' is a refined distributed system, which has the same function as P. Moreover, P' is terminated iff P is terminated, and a process is terminated in P' iff it is terminated in P.

Later, we shall develop termination detectors that each require no more than l overhead messages, where l is the total number of times the processes change state from active to idle. By refining a primitive system as above, the l-value of the system will most likely be reduced and, thus, so is the number of overhead messages required for termination detection.

4. FOUNDATIONS OF THE ALGORITHMS

In the rest of this paper, we consider only process termination detection in a refined system, with the understanding that a primitive system is simply a refined system with no β messages used, and that process termination detection is equivalent to system termination detection when no β messages are used.

Let P be a (refined) distributed system that starts running at time 0. (We assume that the network has no global clock. However, for purpose of presentation, we assume there is a global clock external to the network, and all "times" refer to this external clock.) For any two processes p, q and time $t \geq 0$, define

$sent(p, q, t) =$ number of α-messages sent from p to q before time t;

$received(p, q, t) =$ number of α-messages received by p from q before time t;

$sent(p, t) =$ number of α-messages sent by p before time t, with an x-destination message counted x times;

$received(p, t) =$ number of α-messages received by p before time t;

$deficit(p, t) = sent(p, t) - received(p, t)$.

A distributed system is said to be *nearly terminated* iff all processes are either idle or passive and no α-messages are in transit in the system.

The theorem below shows that in order to detect if a process is terminated, it is first necessary to detect if the system is nearly terminated.

Theorem 1. *A terminated process may recognize itself as terminated only if the system is nearly terminated. Conversely, if the system is nearly terminated and no outstanding β-messages are addressed to a currently idle process, then the process knows itself to be terminated.*

The next two theorems show sufficient conditions for recognizing a nearly terminated system; they form the basis of our termination detectors.

Theorem 2. *Let P be a distributed system. Let t^* and $t_p, p \in P$ be time instants such that $t_p \leq t^*$ for all $p \in P$. P is nearly terminated at time t^* if the following hold.*
(1) No process $p \in P$ is active at time t_p.
(2) $\sum_{p \in P} sent(p, t_p) = \sum_{p \in P} received(p, t_p)$.
(3) There exist no processes $p, q \in P$ such that p sends an α-message after t_p and q receives that message before t_q.

Theorem 3. *Let P be a distributed system. Let t^* and $t_p, p \in P$ be time instants such that $t_p \leq t^*$ for all $p \in P$. P is nearly terminated at time t^* if the following are satisfied.*
(1) No process $p \in P$ is active at time t_p.
(2) $sent(p, q, t_p) = received(q, p, t_q)$ for all $p, q \in P$.

The following corollary is obvious.

Corollary 1. *If the conditions of Theorem 2 (3) hold, then no process p may send any underlying message after time t_p.*

5. ALGORITHM WITHOUT PIGGYBACKING

Let P be a distributed system. From Theorems 1 and 3, we immediately obtain the following algorithm for process termination detection.

Algorithm. Every process $p \in P$ keeps a list $\{S_p(q, r), R(q, r) : q, r \in P\}$ and does the following:

0. Initially, $S_p(q, r) := 0$, $R(q, r) := 1$ for all $q, r \in P$.
1. Whenever p changes state from active to idle, say at time t_p, let p broadcast (to all other processes) a control message carrying the set $\{sent(p, q, t_p), received(p, q, t_p) : q \in P\}$.
2. Whenever p receives from q the control message containing $\{sent(q, r, t_q), received(q, r, t_q) : q \in P\}$, let $S_p(q, r) := sent(q, r, t_q)$ and $R_p(q, r) := received(q, r, t_q)$ for all $r \in P$.
3. Whenever p becomes idle, let $S(p, q) := sent(p, q, t_p)$ and $R(p, q) := received(p, q, t_p)$ for all $q \in P$.
4. Every time step 2 or 3 is executed, check if p is idle and $S_p(q, r) = R_p(q, r)$ for all $q, r \in P, q \neq r$. If yes, then p claims itself to be terminated. □

To show the above algorithm correct, let t_p^* for every $p \in P$ denote the time process p claims itself to be terminated. We show p to be terminated at t_p^*.

Note that process p has received at least one token from every other process by time t_p^*. For $q \neq p$, let msg_q be q's latest control message received by p before time t_p^*, and let t_q be the time msg_q was sent. Let $t_p = t_p^*$. Clearly, the two conditions of Theorem 3 are satisfied relative to these t_p's; therefore the system is nearly terminated at time t_p^*. By Corollary 1, no process q may send any underlying message after time t_q. So, p receives q's underlying messages all before it receives msg_q. By Theorem 1, p is terminated at t_p^*.

On the other hand, one may readily check that once the system nearly terminates, every process will claim itself to be terminated within finite time. This proves the algorithm correct.

Clearly, the total number of control messages used in the algorithm is no more than l, the number of times processes change state from active to idle.

6. ALGORITHM USING PIGGYBACKING

We say that a control message c *piggybacks* on an underlying message u if c and u are merged to a single message with destinations being c's and u's all together. We assume that c and u (including their own destinations) can later be extracted from the merged message by every receiving process. When c is small (say, it contains no data or only one or two bytes of data), the cost of sending $c + u$ is normally far less than the cost of sending c and u separately. Thus by piggybacking control messages on underlying messages, a termination detector may save lots of overhead messages.

The technique of piggybacking is not appealing for the algorithm in the preceding section, for each control message (except the one sent by p_0) carries $\Omega(n)$ pieces of data, where n is the number of processes in the system. This section presents an algorithm that uses only small control messages, which makes piggybacking feasible.

6.1 Description of the Algorithm

This algorithm is based on Theorem 2. It is symmetric in the sense that every process executes the same algorithm. Thus, it suffices to describe the algorithm for an individual process, say, p. (Using notation of Section 2, the algorithm to be described is just $\overline{p}$.) We assume the underlying system P has n processes which are numbered 1 to n; thus, p is an integer satisfying $1 \leq p \leq n$. This assumption enables us to reduce the average-case complexity of the algorithm.

In the algorithm, processes communicate with two kinds of control messages: *signals* and *tokens.* A signal does not carry any data, while each token carries an integer value.

Each process sends a sequence of tokens and signals such that between any two consecutive tokens there is exactly one signal (i.e., tokens alternate with signals). A signal/token is addressed to all processes in the system (except the sender itself); thus, a group name will be sufficient for addressing. A signal and a token (in that order) are occasionally combined into a single message; when such a message is received, the signal is processed prior to the token.

Process p may send a token only when it is idle or passive, or when it is about to become idle (i.e., it becomes idle immediately after the token is sent). Each token carries the following value

$$d(p) = deficit(p, t) - deficit(p, t')$$

where t and t' are respectively the times at which the current token and the previous token are sent ($t' = 0$ if there is no previous token).

After process p has received at least one token from every other process, adding together p's current *deficit* and all the *data* received by p yields

$$D(p) = \sum_{q \in P} deficit(q, t_q)$$

where t_q, for $q \neq p$, is the time the last received token from q was sent, and t_p is the last time p became idle. Clearly, if p is idle and $D(p) = 0$ then both (1) and (2) of Theorem 2 are satisfied. Thus, in order to know if the system is nearly terminated, we only need a mechanism for deciding if condition (3) of Theorem 2 holds.

For this purpose, whenever process p resumes active after it sends a token, let p send a signal to inform all other processes of its becoming active. To save communication cost, the signal is not sent immediately after p resumes active, but instead it is held until p has an underlying message or a token to send, then the signal takes a free ride by piggybacking on the underlying message or token.

To decide the timing of sending tokens, process p maintains a variable $Ntokens(p)$, which is initially set to zero and later incremented (decremented) by one whenever p receives a token (signal).

If process p has an underlying message to send immediately before it changes state from active to idle (such a message is called a *pre-idle* message), then it piggybacks a token on the underlying message. Otherwise, p sends a token only when it is idle/passive and $Ntokens(p) \geq p - 1$. The ideas behind this scheme are explained below.

A token sent by process q and received by p is said to be a *latest* token (from q) if p does not receive any signal from q following that token. At any moment $Ntokens(p)$ indicates the number of latest tokens that p has received. Process p may possibly recognize the system as nearly terminated only if $Ntokens(p) = n - 1$. (If $Ntokens(p) < n - 1$ then it is not known whether all processes are idle/passive.) When $Ntokens(p)$ is small, say $Ntokens(p) < p - 1$, process p presumes that the system is still operating and there is no hurry for p to send a token (to report its status). Indeed, we let it wait until $Ntokens(p) \geq p - 1$. While waiting, p may resume active, in which case p does not need to send any token until p becomes idle again. In this way, lots of tokens might be saved.

The above concepts are spelled out as a set of "rules" in [10].

6.2 Correctness and Complexity

We now establish the correctness of the algorithm by showing (a) that a process may announce itself to be terminated only if the process is really terminated, and (b) that once the system nearly terminates every process will claim itself to be terminated in finite time.

Proposition 1. *If a process claims itself to be terminated, it is really terminated.*

Proposition 2. *Once the system becomes nearly terminated, every process will recognize itself as terminated within a finite amount of time.*

As for message complexity, one readily sees that the algorithm requires at most l tokens and $l - n$ signals, where n is the number of processes in the system and l is the total number of times the processes change state from active to idle. Since signals always piggyback on either underlying messages or tokens, only tokens may cause overhead messages. Thus the algorithm needs at most l overhead messages. Because of the strategy that p does not send a token until $Ntokens(p) \geq p - 1$, the number of overhead messages is normally expected to be less than l.

It is not unusual in distributed computing that a process needs to send a pre-idle message every time it is about to change state from active to idle (e.g., to inform other processes of its new computational results). For such a distributed system, our algorithm requires no overhead message at all.

7. COPING WITH DYNAMIC SYSTEMS

So far we consider only a distributed system in which the set of processes remains fixed for the lifetime of the system. Such a system is called *static*. On the other hand, a *dynamic* system is one in which the process set may vary from time to time, *i.e.*, processes may be created and/or destroyed at any time. We consider in this section the termination detection problem for a dynamic system.

7.1. Model of a Dynamic System

We assume that a dynamic system starts with a set of *starting* processes, which stay alive for the lifetime of the system. A process is either *active*, *passive*, *idle*, or *dead*. A process is *live* if it is not dead. A dead process is one that has been destroyed and no longer exists in the system; however, for our purpose, we still regard destroyed processes as members of the system whose state are dead.

Active, passive, and idle processes obey the same rules as in a static system. In addition, an active process is free to create new processes, one at a time; passive, idle, and dead processes cannot create new processes. Besides, a dead process cannot send or receive a message. A non-starting live process is free to become dead after all messages received are finished.

We assume that no message may be addressed to a yet-to-be-created process (i.e., any destination process of a message must be a member of the system at the time when the message is to be sent). However, a message is allowed to be addressed to a dead process. When a β-message is delivered to a dead process, the message is simply discarded. And when an α-message is delivered to a dead process p, it is forwarded to a pre-assigned process (e.g., p's creator process). If the latter process is also dead, the message is re-forwarded to another process, and so forth, until the message reaches a live process. The bottom line here is that any α-message with x destinations must be received by x processes (probably with repetition). We assume that the forward address of a dead process is kept in the station until the system is detected to be nearly terminated.

For $t \geq 0$, let $P(t)$ denote the process set of the system as of time t (note that $P(t)$ includes dead processes as well as live ones). A dynamic system P is said to be *terminated* at time t iff the following hold at time t:

a) every process in $P(t)$ is either dead or idle;
b) no α-messages are in transit;
c) no β-messages addressed to a live process are still in transit.

P is *nearly terminated* at time t iff no process in $P(t)$ is active and no outstanding message is of type α. A process is *terminated* if it can never be active or passive for the rest of the system's lifetime.

7.2. The Algorithm

Our strategy here is to extend the previous algorithm for static systems, observing that Theorem 2 holds for a dynamic system if condition (2) is modified to

(2′) $\sum_{p \in P(t^*)} sent(p, t_p) = \sum_{p \in P(t^*)} received(p, t_p)$.

The main difficulty here is: How can a process know, at time t^*, if it has received information from *every* other process in $P(t^*)$, while a new process may be created at any moment? In the previous algorithm, updated information about a process is contained in a "latest" token from that process, and process p knows it has received a "latest" token from every other process if $Ntokens(p) = n-1$. This scheme would not work for a dynamic system, for n, the number of processes in the system, is now a variable.

Our approach is to divide the processes into *generations.* All starting processes constitute the 0th generation; and for $i > 0$, a process is in the ith generation iff its creator process belongs to the $(i-1)$st generation. Each process is associated with a generation number gn indicating the generation to which it belongs.

We assume that during the lifetime of the system there are at most k generations of processes in the system. That is, processes of the $(k-1)$-st generation are not allowed to create new processes. This assumption is not essential; it is made for ease of presentation only.

We assume that a predesignated process, say p_0, is in charge of termination detection, and p_0 stays alive during the lifetime of the system. The algorithm for p_0 is different from that for the others.

We first describe the algorithm for each individual process other than p_0. Each process $p \neq p_0$ repeatedly informs p_0 of p's status, and for this it maintains three variables c, d, and $flag$, where d and $flag$ play the same roles as before (see Section 6). Variable c indicates the number of processes that have been created by p since the last time p reported to p_0.

Conceptually, whenever changing state from active to idle or dead, process p reports to process p_0 its values of c and d via a *token*; whenever resuming active, it reports this fact to p_0 through a *signal*; and before p sends any (underlying or control) message, it informs p_0 of its existence by a *notice* (*i.e.*, the first message sent by p must be a notice). Note that each process can send only one notice. The notice, each signal, and each token all carry p's generation number. A token, in addition, carries p's c- and d-values.

For efficiency, notices and signals always piggyback on subsequent underlying messages or tokens. And a token piggybacks on a pre-idle underlying message whenever the latter is available. The pseudocode of the algorithm can be found in [10].

We now describe the algorithm for process p_0, which collects information from all other processes and decides if the system is nearly terminated.

Process p_0 has variables D, $Nprocesses[0..k]$, $Nnotices[0..k]$, $Ntokens[0..k]$. Variable D is used to accumulate all d-values brought in by tokens. For $0 \leq i \leq k$, $Nprocesses[i]$ accumulates the c-values brought in by tokens with generation number i. $Nnotices[i]$ records the number of notices received from processes in generation i. And $Ntokens[i]$ records the number of "latest tokens" received from processes in generation i. In a sense, $Nprocesses[i]$ and $Nnotices[i]$ together indicate p_0's knowledge of the size (population) of generation i.

We will show that when $D = 0$ and $Nnotices = Ntokens = Nprocesses$ (i.e., $Nnotices[i] = Ntokens[i] = Nprocesses[i]$ for every i, $0 \leq i \leq k$), p_0 has a "latest" token from every other process in the system—answering the question raised in the beginning of this section. If in addition p_0 is idle, then the system is nearly terminated and process p_0 is terminated.

After p_0 detects its own termination, it sends a control message to every other process; the latter then declares itself terminated on reception of the control message. The algorithm for p_0 is also spelled out in [10].

7.3. Proof of Correctness

Assume at time t^* that p_0 be idle, $D = 0$, and $Nnotices[0..k] = Ntokens[0..k] = Nprocesses[0..k]$. Let

$$Q(t^*) = \{p_0\} \cup \{q \in P(t^*) : \text{at time } t^*, p_0 \text{ has a "latest" token from } q\}$$

where a token from q is called "latest" as long as p_0 has not received any signal from q following that token.

Let t_{p_0} be the last time p_0 changed state from active to idle. For $q \in Q(t^*)$, $q \neq p_0$, let t_q be the time at which the "latest" token of q was sent. We show below $P(t^*) = Q(t^*)$ by proving a sequence of lemmas. This equality answers the question raised in the beginning of Section 7.2, and once it is established the correctness proof of this algorithm will be similar to that of the previous algorithms.

Lemma 1. *If $p \in P(t^*) - Q(t^*)$, then by time t^* process p_0 has not received any message (underlying or control) from process p.*

As a result of the preceding lemma, we have

$$D(t^*) = \sum_{q \in Q(t^*)} deficit(q, t_q) \tag{1}$$

where $D(t^*)$ is the value of D as of time t^*.

Lemma 2. *Let $p, q \in Q(t^*)$. It cannot happen that p sends an α-message before time t_p and q receives it after time t_q.*

Lemma 3. *If p is q's creator process, where $p, q \in Q(t^*)$, then p creates q before time t_p.*

Lemma 4. $P(t^*) = Q(t^*)$.

The following proposition shows that the algorithm is correct.

Proposition 3. *If a process announces itself to be terminated, it is really terminated. On the other hand, after the system nearly terminates, every process will recognize itself as terminated within a finite amount of time.*

7.4. Reducing Average-Case Complexity

The proposed algorithm requires at most $l - 1$ overhead messages to detect p_0's termination, where l is the number of times the processes change state from active to idle or dead. After p_0 detects its own termination, it requires one more message for the others to detect their termination. Thus, the algorithm totally requires at most l overhead messages.

The average message complexity of the algorithm can be reduced, using the following scheme. Whenever a process sends a signal or token to p_0, let its creator process receive a copy of the signal/token. And, instead of p sending a token immediately after it becomes idle, let p wait until it has received a "latest" token from each of its children. While waiting, p may resumes active and thereby saves a token. This strategy is different from the one used in the algorithm of Section 6. Their motivations are the same, however.

8. LOWER BOUNDS

Each of the foregoing algorithms requires no more than l overhead messages, where l is the number of times the processes change state from active to idle (and/or dead in the case of a dynamic system). We will show in this section that any process termination detector requires l overhead messages in the worst case. Thus, our algorithms are optimal, measured by the number of overhead messages required in the worst case. We will also show that the problem of system termination detection is intrinsically harder than the problem of process termination detection in terms of overhead messages required.

Lemma 5. *Let P be a system of n processes. Suppose at time t^* no messages are in transit and all the n processes change state from active to idle simultaneously. If there is a process, say p, that does not send any overhead message after time t^*, then no process other than p can recognize itself as terminated.*

It follows from Lemma 5 that in order for *every* process to recognize its own termination, every process must send an overhead message after time t^*.

The following theorem shows that every process-termination detector requires at least l overhead messages in the worst case.

Theorem 4. *Let n, l be integers such that $1 < n \leq l$. For any process-termination detector A, there exists a system of n processes with the following properties:*

(1) The processes change state from active to idle or dead for a total number of l times.
(2) In order for every process to recognize itself as terminated, the detector needs at least l overhead messages.

Theorem 5. *Let n, l, m be integers such that $1 < n \leq l$. For any system-termination detector A, there exists a system of n processes with the following properties:*

(1) The processes change state from active to idle or dead for a total number of l times.
(2) The processes send a total number of m β-messages.
(3) Algorithm A requires at least $l + m$ overhead messages in order for every process to detect the system's termination.

The problem of termination detection is usually studied under the assumption that no global clock is available in the network. While this assumption is reasonable in a point-to-point network, it might be too restricted in a broadcast network, especially a local-area network, where clock synchronization is not a big problem. A natural question then arises: With a global clock available, will termination detection become easier? The answer is no. To see this, just observe that no assumption is required about the existence or non-existence of a global clock in the network. Thus, Theorems 4 and 5 still hold even if the network has a global clock.

9. CONCLUSION

We have studied the problem of termination detection in the environment of a broadcast network. We studied the possibility of reducing the complexity of termination detection, and noticed that, in many applications, the problem of termination detection may be formulated as deciding if a process (rather than the entire system) is terminated. We then showed that if users of a termination detector can provide such information as whether a message *msg* requires its receiving process to send another message in execution of *msg*, then a distributed system can be automatically transformed to another (functionally equivalent) system so that the latter normally requires fewer overhead messages for process-termination detection than the former. Such transformation, however, does not help reducing the complexity of system-termination detection.

We also developed message-optimal termination detection algorithms for dynamic as well as static systems; each algorithm requires in the worst case at most l overhead messages, where l is the number of times the processes change state from active to idle or dead. The technique of piggybacking proved to be useful in reducing the number of overhead messages. While these algorithms were developed for process-termination detection in a refined system (static or dynamic), they can be used for system-termination detection in a primitive (static or dynamic) system in which all messages are α-messages. When so used, the algorithms are also message optimal.

REFERENCES

[1] K. M. Chandy and L. Lamport, “Distributed snapshots: determining global states of distributed systems,” *ACM Trans. Comp. Syst. 3* (1985), 63-75.

[2] S. Cohen and D. Lehmann, “Dynamic systems and their distributed termination,” In *Proc. First Annual ACM Symp. on Principles of Distributed Computing*, 29-33, 1982.

[3] E. W. Dijkstra and C. S. Scholoten, “Termination detection for diffusing computations,” *Inform. Processing Lett. 11* (1980), 1-4.

[4] E. W. Dijkstra, W. H. J. Feijen and A. J. M. van Gasteren, “Derivation of a termination detection algorithm for distributed computations,” *Inform. Processing Lett. 16* (1983), 217-219.

[5] N. Francez and M. Rodeh, “Distributed termination,” *ACM Trans. Program. Lang. Syst. 2* (1980), 42-55.

[6] N. Francez, M. Rodeh and M. Sintzoff, “Distributed termination with interval assertions,” In *Proc. Intl. Colloq. Formalization of Programming Concepts* (1981), 280-291, Lecture Notes in Computer Science, Vol. 107, Springer-Verlag.

[7] N. Francez and M. Rodeh, “Achieving distributed termination without freezing,” *IEEE Trans. Software Engineering SE-8* (1982), 287-292.

[8] D. Kumar, “A class of termination detection algorithms for distributed computations,” Technical Report, Dept. Computer Science, University of Texas at Austin, 1985.

[9] T. H. Lai, “Termination detection for dynamically distributed systems with non-first-in-first-out communication,” *Journal of Parallel and Distributed Computing 3* (1986), 577-599.

[10] T. H. Lai and T. H. Yang “Termination detection in a broadcast network,” Tech. Report, Dept. of Computer and Information Science, Ohio State University, 1987.

[11] F. Mattern, “New algorithms for distributed termination detection in asynchronous messages passing systems,” Tech. Report, University of Kaiserslauternm, West Germany, 1985.

[12] J. Misra and K. M. Chandy, "Termination detection of diffusing computation in communicating sequential processes," *ACM Trans. Program. Lang. Syst. 4* (1982), 37-43.

[13] J. Misra, "Detecting termination of distributed computations using markers," In *Proc. Second Annual ACM Symp. on Principles of Distributed Computing*, 290-294, Montreal, August, 1983.

[14] W. Stallings, *Data and Computer Communications*, 1985, Macmillan.

[15] B. Szymanski, Y. Shi and N. Prywes, "Synchronized Distributed Termination," *IEEE Trans. Software Engineering SE-11* (1985), 1136-1140.

[16] R. Topor, "Termination detection for distributed computations," *Inform. Processing Lett. 18* (1984), 33-36.

ON THE PERFORMANCE OF SW-BANYAN NETWORKS

M.H. PEYRAVI
Department of Mathematical Sciences
Kent State University
Kent, Ohio 44242

M.R. MEYBODI
Department of Computer Science
Ohio University
Athens, Ohio 45701

ABSTRACT

A class of interconnection networks is studied as a communication medium in multi-processing systems. These networks, termed expanding and contracting SW-banyan networks, recently introduced by DeGroot. In this paper a new method for computing the number of permutations realized by this class is given. It is shown that the known method for computing this number can not be used for any SW-banyan network. Finally, we study different classes of expanding and contracting NxN SW-banyan networks with respect to three different performance measures: number of links, number of switches, and blockage of the network. We derive conditions on fanout and spread vectors of SW-banayan networks to minimize one or more of these quantities.

Index terms: *banyan networks, SW-banyan, blockage, combinatorial power.*

I. INTRODUCTION

A fundamental class of interconnection networks, in the context of multi- processing systems, is called banyan networks and were first proposed by Goke and Lipovski[1]. The banyan class contains a very rich and useful subclass of networks called L-stage banyan networks [1],[2]. A variety of special cases of L-stage banyan networks have received considerable attention in the design and in the literature. Large banyan networks can be synthesized from smaller ones. This is illustrated in Figure 1(a). The interconnection of these banyan networks can be represented by a switching network as in Figure 1(b). A cross-bar switch is a trivial one level banyan network. L-stage banyan networks are synthesized recursively from cross-bar switches as in Figure 1(b). Two of the most notable subclasses of an L-stage banyan are Delta networks [4],[5],[6], and SW-banyan networks. The well known networks such as Omega networks[7], Indirect Binary n-Cube networks[8], STARAN flip networks[9] and Baseline networks[10] are all special cases of Delta networks. The characteristic property of a Delta network is a simple decentralized routing algorithm based on "destination tag". The SW-banyan class is a proper subset of banyan networks that can be constructed recursively from smaller SW-banyans. Another interesting class of SW-banyan networks is *Expanding and Contracting* SW-banyan networks which have been studied in [2],[11],[12],[13].

In this paper we study the performance of expanding and contracting SW-banyan networks with respect to four parameters, namely, number of permutations realizable, number of links, number of switches and number of blockage in the network.

In section II different classes of banyan networks are described. In section III, we compute the number of permutations realized by a given L-stage expanding and contracting SW-banyan network. We show that the method due to Bhuyan and Agrawal [14] for computing the number of permutations realized is incorrect. We introduce a new method for computing this quantity. Finally, a trade off between path blockages and combinatorial power of a network is discussed. In section IV different classes of expanding and contracting NxN SW-banyan networks are analyzed with respect to different performance measures such as number of links, number of switches and blockage in the network. We find conditions on elements of the fanout vector and spread vector such that one or more of these parameters are minimized.

II. BANYAN NETWORKS

A banyan graph is a Hasse diagram of a partial ordering in which there is only one path from any base to any apex. A base is defined as any vertex with no arcs incident into it and an apex is defined as any vertex with no arcs incident out of it, and all other vertices are called intermediates. In a computer system, the set of bases may correspond to processors, the set of apexes to memory and I/O devices and the intermediates to switch nodes. One of the useful property of a banyan network is that there is one and only one path from any base to any apex (unique path network).

An L-level banyan network is a banyan graph in which the path from base to apex (or apex to base) is of length L. Therefore, in an L-level banyan, there are L+1 level of nodes and L-level of edges. An L-level banyan is an (L+1)-partite graph in which the partitions may be linearly ordered from 0 though L such that arcs exist only from i^{th} to $(i+1)^{\text{th}}$ partition. By convention, apexes(outputs) are considered to be at level 0 and bases(inputs) are at level L.

Definition 1: In a banyan network, the spread of a vertex is the out-degree of a node and the fanout of a vertex is the in-degree of a node (the direction is from base to apex).

Definition 2: If all vertices within the same level of a banyan network have identical spread and fanout value then the banyan is called uniform, Figure 2, otherwise it is called non-uniform.

In a uniform banyan network, the fanout values and the spread values may be characterized by L component vectors, $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$, the fanout vector and spread vector, respectively, where s_i and f_i denote the spread and fanout of a node at level i. Note that $s_i \times f_{i-1}$ is the size of the switches at level i, for $1 \leq i \leq L$ and $f_L = s_0 = 0$.

Definition 3: If $s_{i+1} = f_i$, $0 \leq i \leq L-1$, that is $S = F$ then the banyan network is called rectangular, Figure 3. If $s_{i+1} \neq f_i$, for some $0 \leq i \leq L-1$, then it is called non-rectangular.

Definition 4: If every component of S is equal to some constant s and every component of F is equal to some constant f then the banyan is called regular, Figure 4. otherwise it is irregular.

If $s = f$ this implies that F=S and the banyan network is both regular and rectangular which is called strongly rectangular.

SW-Banyan Networks:

Let A be the set of apexes (output terminals) and B be the set of bases (input terminals). Given $b \in B$ and $a \in A$, define $R_x(a)$, (x reachability set for apex a), to be a set of all nodes at level x, $(0 \leq x \leq L)$, which are reachable from apex a, and $R_x(b)$, (x reachability set for base b), to be a set of all nodes at level x, $(0 < x < L)$, which are reachable from base b. Clearly,

$$\begin{aligned} R_0(a) &= \{a\} \\ R_L(a) &= B, && \text{the set of input terminals (bases).} \\ R_0(b) &= A, && \text{the set of output terminals (apexes).} \\ R_L(b) &= \{b\} \end{aligned}$$

Definition 5: An L-stage uniform banyan is called SW-banyan if and only if either one or both of the following conditions is true.

$$\begin{array}{ll} 1) & R_x(a_i) = R_x(a_j) \quad \text{or} \quad R_x(a_i) \bigcap R_x(a_j) = \phi \\ 2) & R_x(b_i) = R_x(b_j) \quad \text{or} \quad R_x(b_i) \bigcap R_x(b_j) = \phi \end{array}$$

for $a_i, a_j \in A, b_i, b_j \in B$ and $0 \leq x \leq L$. In other words, A banyan network is SW-banyan if and only if for any two bases b_1 and b_2 (or any two apexes a_1 and a_2, their reachability sets at each level are disjoint or identical.

If an L-stage uniform banyan network satisfies either 1) or 2) but not both then it is called one-way SW-banyan network. If condition (1) is true then it is called input SW-banyan and if condition (2) is true then it is called output SW-banyan. If both 1) and 2) are true, then it is called two-way SW-banyan.

From the definition of the L-stage uniform NxM banyan networks it follows [2] that

$$N = \prod_{i=1}^{L} s_i \qquad M = \prod_{i=0}^{L-1} f_i.$$

Given this factorization for N and M, an L-stage input SW-banyan network is defined recursively as follows: The input stage (stage L) is made of N/s_L cross-bar switches, each of size $s_L * f_{L-1}$. This stage is followed by f_{L-1} blocks. Each of these blocks is an (L-1)-stage block structured SW-Banyan network and its input stage contains $N/s_L s_{L-1}$ cross-bar switches of size $s_{L-1} \times f_{L-2}$. The recursion stops at stage 1. The output terminal of switches in stage i are connected to the input terminals of blocks in stage $i+1$. Below we state some theorems and corollaries which discuss topological feature of SW-banyan networks [2].

Theorem 1: For any level i in an SW-banyan $0 \leq i \leq L-1$, $n_{i+1}s_{i+1} = n_i f_i$, where n_i is the number of nodes at level i.

Theorem 2: For any uniform L-level SW-banyan $n_i < n_{i+1}$ if and only if $f_i > s_{i+1}$. Furthermore, $n_i > n_{i+1}$ if and only if $f_i < s_{i+1}$, and $n_i = n_{i+1}$ if and only if $f_i > s_{i+1}$.

Theorem 3: In a rectangular SW-banyan n_i is constant and equal to $|B|$, the number of bases in the banyan, for $0 \leq i \leq L$.

Corollary 3.1: If $n_i = n_{i+1}$ for all i, $0 \leq i \leq L-1$, then $F = S$.

Theorem 4: Each base of an L-level SW-banyan reaches $s_{i+1}s_{i+2}\ldots s_L$ nodes at level i, $0 \leq i \leq L-1$.

Corollary 4.1: The number of apexes in a uniform SW-banyan is $s_0 s_1 s_2 \ldots s_L$

Corollary 4.2: Each node at level i, $0 \leq i \leq L$, reaches $s_0 s_1 s_2 \ldots s_i$ apexes.

Theorem 5: Every apex in a uniform L-level SW-banyan reaches $f_0 f_1 \ldots f_{i-1}$ nodes at level i, $0 \leq i \leq L$.

Corollary 5.1: The number of bases in a uniform SW-banyan is $f_0 f_1 f_2 \ldots f_L$.

Corollary 5.2: Each node at level i, $0 \leq i \leq L$, reaches $f_i f_{i+1} \ldots f_L$ bases.

Theorem 6: In a SW-banyan network, the number of apexes equals the number of bases if and only if $f_0 f_1 \ldots f_{L-1} = s_1 s_2 \ldots s_L$.

Expanding and Contracting NxN SW-banyan Networks:

Consider a non-prime integer $N > 2$. Let $N = n_1 n_2 \ldots n_L$ be a factorization of N, where all factors are not necessarily distinct. Let $n = (n_1, n_2, \ldots, n_L)$ be a vector of the L factors. Let $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$ be two not necessarily distinct permutations of the components of n. Given S and F, consider an L-stage SW-banyan network where the i^{th} stage is made up of complete cross-bar switches of size $s_i \times f_{i-1}$ [1,2]. Clearly, the number of switches in stage 1 is N/s_L and in stage i is given by $N s_1 s_2 \ldots s_{i-1}/f_1 f_2 \ldots f_i$ for $1 \leq i \leq L$. Let r be the number of distinct permutations of the L factors of N, then there are r^2 such networks. By exhausting the set of all factors of N, a class of all SW-banyan networks with N inputs and N outputs can be obtained.

Example: If $N = 2 \times 2 \times 4$ then there are three distinct permutations, $(2,2,4)$, $(4,2,2)$ and $(2,4,2)$ and hence there are nine networks. For $N = 16$ there are 16 different SW-banyan networks including the 16×16 cross-bar switch which are illustrated in Table 1. This class of networks includes all the well-known Delta networks such as Omega networks, indirect binary networks as well as the class of expanding and contracting SW-banyan networks [2, 5].

If $F = S$ that is $s_i = f_{i-1}$ for $1 \leq i \leq L$, then each stage is made of square cross-bar switches. If $S \neq F$, that is there exist $j, 1 \leq j \leq L$ such that $s_j \neq f_j$, then there is at least one stage made of rectangular cross-bar switch. For example, consider $F = (2,2,4)$ and $S = (2,4,2)$, then stage one is made of 2×2 switches and the second the third stages are made of switches of size 2×4 and 4×2, respectively, Figure 5. If $S \neq F$, the network is called Expanding-Contracting SW-banyan Network [2].

It is shown through examples [2] that Expanding and Contracting SW-banyan networks have less path blockage, i.e., when one base is connected to one apex by a direct communication path, no other new base-apex connection can be made if the new connection requires a link in use by the first connection, the second connection is blocked by the first one. As a result less blockage could imply more permutations to be realized by the network. Table 1 illustrates the blockage for the set of all 16x16 SW-banyan networks.

III. NUMBER OF PERMUTATIONS REALIZED BY SW-BANYAN NETWORKS

While there is a reason to believe that networks with less blockage may realize more permutations, computing the latter quantity for the expanding and contracting SW-banyan networks is by no means trivial and in fact is equivalent to a class of enumeration problems which are of independent interest. Two different cases will be analyzed, one for $S = F$ and the other for $S \neq F$.

Case A: If $F = S$, then all the switches of each stage have the same size, i.e., $(s_i \times s_i), 1 \leq i \leq L$, where L is the number of stages of the network. Each switch is capable of realizing $s_i!$, then the number of permutations $\alpha(N)$ realized by an $N \times N$ SW-banyan network is given by:

$$\alpha(N) = \prod_{i=1}^{L} (s_i!)^{m_i}$$

where m_i is the number of switches in stage i. The instances 2, 3, 5, 8, 12, 15 and 16 in Table 1 correspond to this case.

Case B: If $F \neq S$, then it is not always necessary that $\alpha(N) \geq 0$. For example if $F = (8,2)$ and $S = (2,8)$, then stage 1 is made of two copies of 2×8 switches and stage 2 contains two copies of 8×2 switches. Figure 6 illustrate the network. Since the output of stage one contains only four output ports, 12 of the 16 inputs are blocked by stage 1 and hence $\alpha(N) \geq 0$ Consequently, the following theorem is immediate.

Theorem 7: Let the j^{th} stage of an L-stage $N \times N$ SW-banyan network is made of cross-bar switches of size $s_j \times f_{j-1}$, $1 \leq j \leq L$. Then a necessary and sufficient condition for $\alpha(N) \geq 0$ is that $m_j s_j \geq N$, where m_j is the number of switches in stage j.

The above condition is equivalent to the following statement:

$$s_1 s_2 \ldots s_j \geq f_1 f_2 \ldots f_j$$

It is easy to see that $\alpha(N) = 0$ for instances 6,9,13 and 14. The remaining four instances namely 4,7,10 and 11 satisfy the condition of the above theorem. The computation of $\alpha(N)$ for later instances of the SW-banyan network gives rise to an interesting class of enumeration problems. We illustrate using the example in Figure 5, where $F = (2,2,4)$ and $S = (2,4,2)$. Consider the first block which consists of switches A_1 to A_4 and B_1 to B_4. The structure of the network induces the following natural constraints. Each of the switches $A_i (1 \leq i \leq 4)$ has four output ports, since there are only two input ports to A_i, only two out of four output ports can actually carry the two inputs. Likewise, each switch $B_j (1 \leq j \leq 4)$ through the link permutation can receive four inputs, since it has only two output links. To avoid the blockage within the switches $B_j (1 \leq j \leq 4)$, it is required that B_j receives exactly two inputs from any two of the four input ports. A similar requirement holds for the switches in the second block. In other words, each switch in stage 2 and 3 receives exactly two inputs and acts as though they were 2×2 switches. Thus, within the first block, the number of permutations realized by the network crucially depend on the number of ways in which each of the switches A_i sends its two outputs to the switches B_j under the constraint that each B_j can receive exactly two inputs. This number, as it can be seen, is the same as the number of distinct 4×4 matrices where the elements of the matrices belong to the set $\{0, 1\}$ and the sum of each row and the sum of each column is equal to 2.

The following matrix corresponds to the first (second) block,

$$\begin{array}{ccccc} & A_1 & A_2 & A_3 & A_4 \\ B_1 & 1 & 0 & 1 & 0 \\ B_1 & 0 & 1 & 0 & 1 \\ B_1 & 1 & 1 & 0 & 0 \\ B_1 & 0 & 0 & 1 & 1 \end{array}$$

where the i^{th} row corresponds to the i^{th} switch B_i and the j^{th} column corresponds to the j^{th} switch A_j. The $(i, j)^{\text{th}}$ element is 1 if and only if B_i receives an input from A_j.

Let $f(a, A, b)$ be the number of distinct matrices whose elements belong to the set A and every row and column sum is b. For the above matrix, this number is $f(4, \{0, 1\}, 2)$. It can be easily seen that this number is 90. Notice the first block and the second block of Figure 5(b) can be controlled independently. With this constraint, each of the 4×2 and 2×4 switches acts as a 2×2 switch only. Thus each of the switches at stage 2, and 3 realizes two permutations. There are 24 switches involved in the network; therefore the number of permutations realized by the network is:

$$\alpha(N) = 2^{24} f^2(4,\ \{0,1\}, 2).$$

Referring to Table 1, the network corresponding to the instance 4 and 13 give rise to the computation of $f(8, 0, 1, 2)$ and $f(4, 0, 1, 4)$, respectively. These later enumeration problems are of intrinsic interest.

Table 1 illustrates $\alpha(N)$, path blockage [2] and the cost of each network. The cost of a network is measured in terms of the total number of switching elements.

It follows that less blockage means more combinatorial power measured in terms of the number of permutations realized. Further, among the networks with the same cost (instances 2, 8, 12, 15, and 16) instance 2 has the minimum number of blockages and maximum value of $\alpha(N)$ and instance 16 has maximum number of blockage and minimum value for $\alpha(N)$. Another fact that emerges from this analysis is that among the networks with the same cost, those with the larger switches have less blockage and larger value for $\alpha(N)$. Another immediate consequence of our analysis is that the formula for the number of permutations given in Theorem 3 of [14] is incorrect, as the following example illustrates.

Example: Consider the network of block A of Figure 5(b). It consists of two stages of 2×4 and 4×2 switches, respectively. Bhuyan and Agrawal [14] showed that the number of permutations realized by a generalized interconnection network can be obtained by

$$P = \prod_{i=1}^{r} S_i^{k^i}$$

where k_i is the number of switches at the i^{th} stage and

$$S_i = \binom{n_i}{m_i} m_i \text{ for } m_i \leq n_i$$

$$S_i = \binom{m_i}{n_i} n_i \text{ for } m_i \geq n_i$$

where S_i is the number of permutations achievable by an $m_i \times n_i$ cross-bar switch at the i^{th} stage and r is the number of stages of the network.

For the network of block A of Figure 5(b), $m_1 = 2, n_1 = 4, m_2 = 4, n_2 = 2, r = 2$ and $k^i = 4$, for $i = 1, 2$ then

$$S_1 = \binom{4}{2} 2! = 12$$

$$S_2 = \binom{4}{2} 2! = 12$$

$$P = \prod_{i=1}^{r} S_1^{k_i} = (12)^4 (12)^4 = 12^8$$

while this number is computed to be 90 according to the method discussed earlier.

IV. OPTIMAL EXPANDING AND CONTRACTING NxN SW-BANYAN NETWORKS

In this section we study different classes of Expanding and Contracting $N \times N$ SW-banyan with respect to three different performance measures: number of links, number of switches and blockage in the network. We find conditions on elements of vector F and S such that one or more of these parameters are minimized. We use function $BP(i) = (a_i - 1)(b_i - 1)$ suggested in [2] to measure the amount of blockage in the network. In this equation, b_i is the number of bases reachable by a node at level i and a_i is the number of apexes reachable by a node at level i. This function counts the number of blocked paths that pass through a busy node at level i. The sum of all $BP(i)$ for $1 \leq i \leq L-1$ which is denoted by TDP can be used to measure the blockage in the network. Both $BP(0)$ and $BP(L)$ are defined to be zero. In the above sum, many blocked paths would be counted more than once and so they would not give the total number of blocked paths generated by a single active path. Function $BL(i) = (a_i - a_{i-1})(b_i - 1)$ avoids multiple blocked paths that are encountered. This function counts the number of additional blocked paths that are encountered as a communication path is followed from an apex down to a base. The sum $TBL = \sum_{i=0}^{L-1} BL(i)$ gives the total blockage created by a single connection.

Theorem 8: Total number of nodes in an SW-banyan graph with $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$ is

$$TN = n_0 \left(1 + \sum_{i=0}^{L-1} \frac{\prod_{j=0}^{i} f_j}{\prod_{j=1}^{i+1} S_j} \right)$$

Theorem 9: Total number of edges in an SW-banyan graph with $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$ is

$$TE = n_0 f_0 \left(1 + \sum_{i=0}^{L-1} \frac{\prod_{j=1}^{i+1} f_j}{\prod_{j=1}^{i+1} S_j} \right)$$

Theorem 10: Total number of switches in an SW-banyan network corresponding to the SW-banyan graph with $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$ is

$$TS = n_0 \sum_{i=0}^{L-1} \frac{\prod_{j=0}^{i-1} f_j}{\prod_{j=1}^{i+1} S_{j+1}}$$

Theorem 11: Total number of links in an SW-banyan network corresponding to the SW-banyan graph with $F = (f_0, f_1, \ldots, f_{L-1})$ and $S = (s_1, s_2, \ldots, s_L)$ is

$$TL = n_0 \left(\sum_{i=0}^{L-2} \frac{\prod_{j=0}^{1} f_j}{\prod_{j=0}^{i} S_{j+1}} \right)$$

Theorem 12: The amount of blockage in an SW-banyan network is inversely proportional to the number of switches

$$TBL = \frac{n_0}{TS} \left(\sum_{i=0}^{L-1} \left(\frac{s_0}{s_{i+1}} - \frac{s_0}{s_i s_{i+1}} \right) + \sum_{i=0}^{L-1} \prod_{j=0}^{i} s_{i+1}(1 - s_i) \right)$$

Class 1: Let $H_{F,S,N,L}$ denotes the set of all $N \times N$ L-level SW-banyan networks which can be derived by permuting the components of F or S (not both). Clearly, $N = \prod_{i=1}^{L} s_i = \prod_{i=0}^{L-1} f_i$. Figure 2 illustrates an element of set H where $F = (2, 2, 4)$ and $S = (2, 2, 4)$. Consider Figure 5, this SW-banayn is obtained by exchanging the second and the last component of vector S. Let $h_{F_i, S_i, N, L}$ where $F_i = (f_{i0}, f_{i1}, \ldots, f_{i(L-1)})$ and $S_i = (s_{i1}, s_{12}, \ldots, s_{iL})$ denote the i^{th} member of set H. We state the following theorems for class H.

Theorem 13: The banyan network $h_{F_i,S_i,N,L} \in H_{F,S,N,L}$ has minimum number of switches if and only if $(f_{i0} < f_{i1} < \ldots < f_{i(L-1)})$ and $(s_{i1} > s_{i2} > \ldots > s_{iL})$.

Theorem 14: The banyan network $h_{F_i,S_i,N,L} \in H_{F,S,N,L}$ has minimum number of links if and only if $(f_{10} < f_{i1} < \ldots < f_{i(L-1)})$ and $(s_{i1} > s_{i2} > \ldots > s_{iL})$.

Theorem 15: The banyan network $h_{F_i,S_i,N,L} \in H_{F,S,N,L}$ has minimum TBP if and only if $(f_{i0} < f_{i1} < \ldots < f_{i(L-1)})$ and $(s_{i1} > s_{i2} > \ldots > s_{iL})$.

Theorem 16: The banyan network $|H_{F,S,N,L}| = \frac{(L!)^2}{\prod_{i=1}^{L}(g_i)!\prod_{i=1}^{L}(l_i)!}$, where g_i is the number of times that f_i is repeated in F and l_i is the number of times that s_i is repeated in S. $|H|$ denotes the cardinality of set H.

Problem of finding $h_{F_i,S_i,N,L}$ for which TBL is minimum can be reduced to minimizing

$$\sum_{i=0}^{L-1}(s_0s_1\ldots s_i - s_0s_1\ldots s_{i-1})(f_{i+1}f_{1+2}\ldots f_{L-1})$$

subject to the following constraints:

1. $F \in \{F_1, F_2, \ldots, F_x\}$
2. $S \in \{S_1, S_2, \ldots, S_y\}$

where $x = \frac{(L!)}{\prod_{i=1}^{L}(g_i)!}$ and $y = \frac{(L!)}{\prod_{i=1}^{L}(l_i)!}$. Non linear programming may be used to solve the above minimization problem.

Class 2: Let $H^*_{N,L}$ denotes the set of all $N \times N$ L-level SW-banyan networks for which no node has a unit fanout and spread value and let $P = \{P_1, P_2, \ldots, P_r\}$ be a set whose elements represent different ways of factoring N into L integers. Each set P_i has L elements, $P_i = \{P_{i_1}, P_{i_2}, \ldots, P_{i_L}\}$ and the product of these elements is N. Set P will be singleton if N is either a prime number or can be factored into L prime uniquely. We assume $P_{ij} \neq 1$, for all $1 \leq i \leq r$ and $1 \leq j \leq L$. We also assume that no member of set P can be derived by permuting the member of other members of set P. An L-level $N \times N$ SW-banyan can be obtained by choosing a pair of elements from set P and using them as fanout and spread vectors, therefore

$$H^*_{N,L} = \bigcup_{i=1}^{r}\bigcup_{j=1}^{r} H_{F_{P_i}, S_{P_j}, N, L}$$

where F_{P_i} and S_{P_i} are fanout and spread vectors that correspond to P_i and P_j, respectively.

Theorem 17:

$$|H^*_{N,L}| = (L!)^2\sum_{j=0}^{r}\sum_{i=0}^{r}\frac{1}{\prod_{i=1}^{L}(g_{ik})!\prod_{i=1}^{L}(g_{jk})!},$$

where g_{ij} is the number of times that P_{ij} is repeated in set P_i and $r = |P|$.

Based on theorem 13 we propose the following procedure for finding an L-level NxN SW-banyan network that has minimum TC and it is also a member of $H^*_{N,L}$.

Step 1: Compute set P for integer N.

Step 2: For all possible pairs of elements from set P do

a. Sort the elements of the first set in the pair into increasing order.
b. Sort the elements of the second set in the pair into decreasing order. (call the set of all sorted pairs X)

Step 3: Choose a pair from set X that minimize TC.

Step 4: Use the first element of the pair found in step 3 as fanout vector and second element as spread vector.

A similar procedure can be written to find a network with minimum TS or TBP.

Class 3: Let $H'_{F,S,N,L}$ denotes the set of all networks that can be obtained by permuting elements of F or S (or both) and F= Perm(S) [F=Perm(S) means that vector F can be obtained by permuting the elements of vector (S)]. For the set $H'_{F,S,N,L}$ we can state the following theorems.

Theorem 18: $|H'_{F,S,N,L}| = \left(\frac{L!}{\prod_{i=1}^{L}(g_i)!}\right)^2$ where g_i is the number of times that f_i is repeated in the set F (or S).

Theorem 19: Set $H'_{F,S,N,L}$ contains $\frac{L!}{\prod_{i=1}^{L}(g_i)!}$ rectangular SW- banyan networks.

Theorem 20: A network belonging to set $H'_{F,S,N,L}$ has minimum TS and TL if and only if $f_0 < f_1 < \ldots < f_{L-1}$ and $s_1 > s_2 > \ldots > s_L$.

Theorem 21: A network belonging to set $H'_{F,S,N,L}$ has minimum TBP if and only if $f_0 > f_1 > \ldots > f_{L-1}$ and $s_1 < s_2 < \ldots < s_L$.

V. CONCLUSION

Different classes of Expanding and Contracting NxN SW-banyan have been studied with respect to the number of permutations realized, number of switches involved, number of arcs, number of nodes and blockage in the network. The main result of this report consists in computing the number of permutations realized by the SW-banyan networks. It is shown that the known method for computing this number is incorrect and the correct method is given. This method gives rise to an interesting class of enumeration problems. Necessary and sufficient conditions are given to minimize one or more of these parameters.

TABLE 1

PROPERTIES OF SW-BANYAN NETWORKS WITH N=16.

Instances	F	S	Number of Switching Elements	Path Blockage	$\alpha(N)$, Number of Permutation Realized
1	(16)	(16)	256	0	16!
2	(4,4)	(4,4)	128	9	$2^{24} \times 3^8$
3	(2,8)	(2,8)	160	7	$(8!)^2 \times$ $(2!)^8 =$ $(2^{22} \times 992, 25$
4	(2,8)	(8,2)	256	1	$2^{16} \times$ $f(8, \{0,1\}, 2),$ $= 187530840$
5	(8,2)	(8,2)	160	7	$(8!)^2 \times$ $(2!)^8 =$ $2^{22} \times 992, 25$
6	(8,2)	(2,8)	64	49	0
7	(2,4,2)	(4,2,2)	160	9	$2^{24} \times f^2$ $f(4, \{0,1\}, 2),$ $= 90$
8	(2,4,2)	(2,4,2)	128	13	$2^{16} \times 4! =$ $2^{28} \times 3^4$
9	(2,4,2)	(2,2,4)	96	25	0
10	(2,2,4)	(2,4,2)	160	9	$2^{24} \times f^2$ $f(4, \{0,1\}, 2),$ $= 90$
11	(2,2,4)	(4,2,2)	192	5	$2^{16} \times$ $f(4, \{0,2,2\}, 4)$
12	(2,2,4)	(2,2,4)	128	13	$2^{28} \times 3^4$
13	(4,2,2)	(2,2,4)	80	33	0
14	(4,2,2)	(2,4,2)	96	25	0
15	(4,2,2)	(4,2,2)	128	13	$2^{28} \times 3^4$
16	(2,2,2,2)	(2,2,2,2)	128	17	2^{32}

REFERENCES

[1] Goke, L. R. and Lipovski, G. J., 1973, "Banyan Networks for Partitioning Multiprocessor Systems," *Proceedings of the First Annual Symposium on Computer Architecture*, pp. 21-28.

[2] DeGroot, D., 1983, "Expanding and Contracting SW-banyan Networks," *Proceedings of the International Conference on Parallel Processing*, pp. 771-780.

[3] Goke, L. R., 1976, "Banyan Networks for Multiprocessor Systems," *Ph.D. Dissertation, University of Florida.*

[4] Dias, D. M., 1981, "Analysis and Simulation of Buffered Delta Networks," *IEEE Transactions on Computers,* Vol. 30, pp. 273-282.

[5] Kruskel, C. P. and Snir, M., 1982, "Some Results on Multi-stage Interconnection Networks for Multiprocessors," *Proceedings of the Princeton Conference on Information Sciences and Systems*, pp. 305-310.

[6] Patel, J., 1981, "Performance of Processor Memory Interconnections for Multiprocessors," *IEEE Transactions on Computers*, Vol. 30, pp. 252-264.

[7] Lawrie, D. H., 1975, "Access and Alignment of Data in an Array Processor," *IEEE Transactions on Computers*, Vol. 24, pp. 1145-1155.

[8] Pease, M. C., 1977, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Transactions on Computers*, Vol. 26, pp. 250-265.

[9] Batcher, K. E., 1976, "The Flip Network in STARAN," *Proceedings of the 1976 International Conference on Parallel Processing*, pp. R65-71.

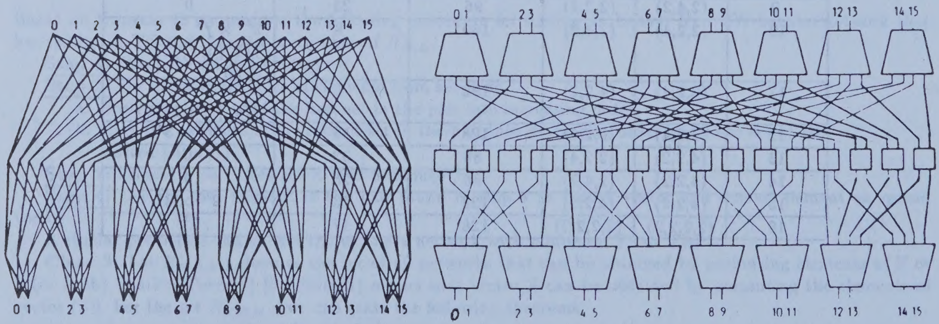
[10] Wu, C. and Feng, T., 1980, "On a Class of Multistage Interconnection Networks," *IEEE Transactions on Computers*, Vol. 29, pp. 694-702.

[11] Meybodi, M. R., 1985, "Optimal Expanding and Contracting nxn SW-banyan Networks," *Proceedings of the Sixteenth Annual Pittsburgh Conference*, pp. 633-638.

[12] Peyravi, M. H., 1985, "Study of Interconnection Networks," *Ph.D. Dissertation, University of Oklahoma, Norman, Oklahoma.*

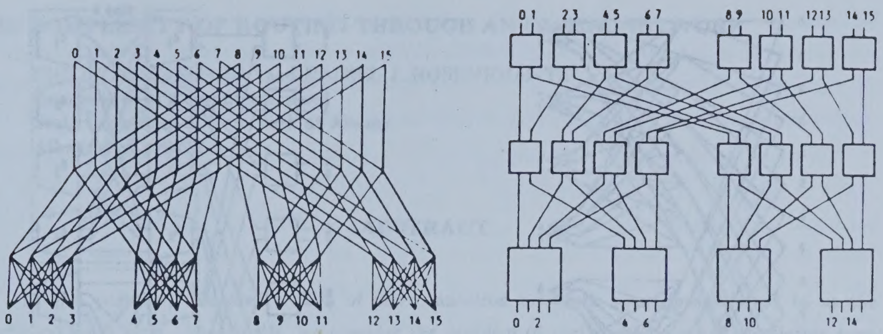
[13] Kothari, S., Lakshmivarahan, S., and Peyravi, M. H., 1984, "Computing the Number of Permutations Realizable by a Class of SW-banyan Networks," *Proceedings of the 1984 ACM Regional Conference, Austin, Texas.*

[14] Bhuyan, L. N. and Agrawal D. P., 1983, "Design and Performance of Generalized Interconnection Networks," *IEEE Transactions on Computers*, Vol. 32, pp. 1081-1090.



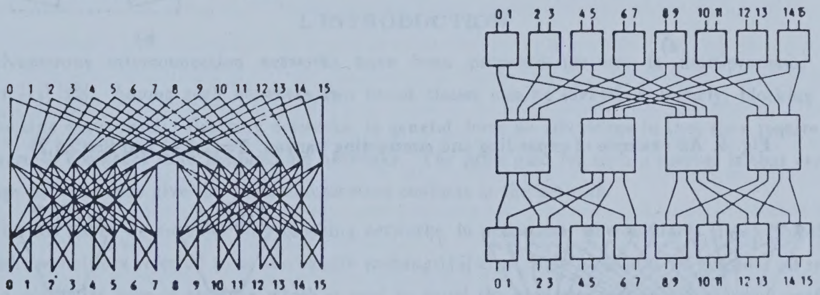
a) b)

Fig. 1. An example of L-level banyan. F=(2,2,4) and S=(4,2,2).


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 4 6 8 10 12 14

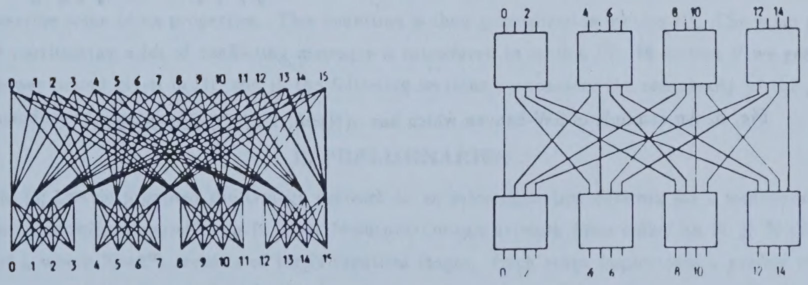
a) b)

Fig. 2. An example of uniform banyan. F=(2,2,4) and S=(2,2,4)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

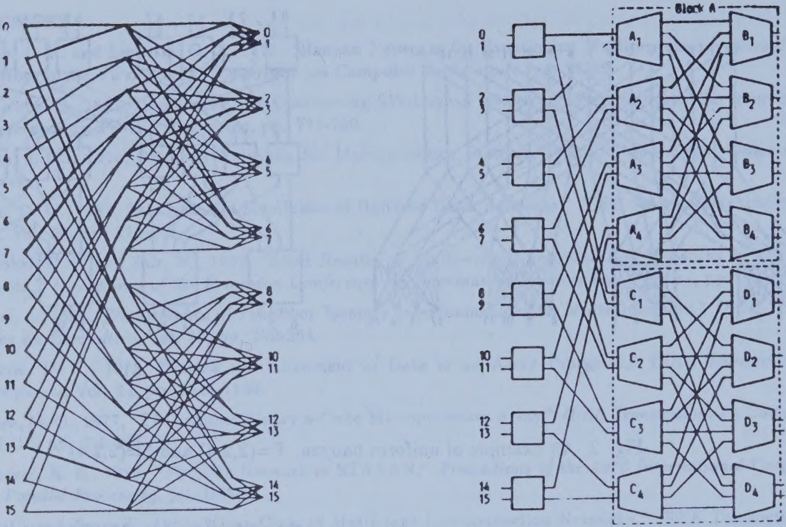
a) b)

Fig. 3. An example of rectangular banyan. F=(2,4,2) and S=(2,4,2)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 4 6 8 10 12 14
0 2 4 6 8 10 12 14

a) b)

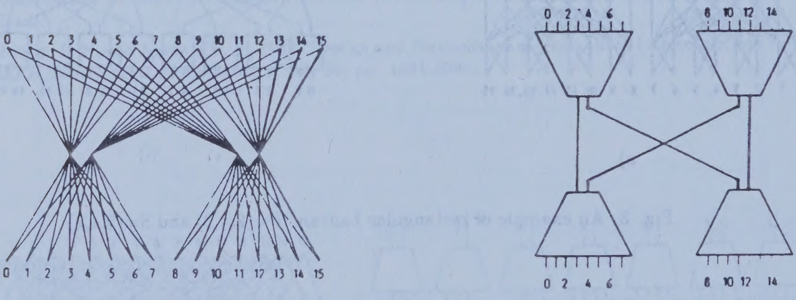
Fig. 4. An example of regular banyan. F=(4,4) and S=(4,4)


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Block A
A1
A2
A3
A4
B1
B2
B3
B4
C1
C2
C3
C4
D1
D2
D3
D4

a)

b)

Fig. 5. An example of expanding and contracting banyan. F=(2,2,4) and S=(2,4,2)


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 4 6
8 10 12 14
0 2 4 6
8 10 12 14

a)

b)

Fig. 6. An example of SW-banyan which has $\alpha(N)=0$. F=(8,2) and S=(2,8)

THE COMPLEXITY OF ROUTING THROUGH AN OMEGA NETWORK

PHILIP J. BERNHARD [†] and DANIEL J. ROSENKRANTZ[‡]
Department of Computer Science
State University of New York at Albany
Albany N.Y. 12222

ABSTRACT

In this paper we consider a class of interconnection networks sometimes referred to as the *shuffle exchange* class. Specifically, we consider the problem of partitioning a set of conflicting messages into a minimum number of subsets, called *rounds*, each free of communication conflicts. We also consider the above problem for a more general class of networks where the interconnection links in the network are replaced by bundles of wires. For a number of special cases we show the above problem to be NP-complete; we show others to be solvable in polynomial time. We also generalize a known lower bound on the number of rounds required by a given set of messages.

I. INTRODUCTION

Numerous interconnection networks have been proposed for use in multiprocessor systems[4],[11],[12]. Among such networks two broad classes can be identified, namely, blocking and nonblocking networks[2]. Blocking networks, in general, have an advantage in that they require less hardware to construct than nonblocking networks. The price paid for such a savings is that certain message patterns may give rise to communication conflicts in the network.

In this paper we consider only blocking networks, in particular, we consider a class of blocking networks sometimes referred to as the *shuffle exchange*[15] class. The problems we consider all result from a particular type of strategy which is used to avoid the occurrence of communication conflicts in such networks. Specifically, we consider the problem of partitioning a set of conflicting messages into a minimum number subsets, each free of communication conflicts. Though we focus on one member of the shuffle exchange class, namely the Omega network[8], our results hold for all networks in that class.

The remainder of this paper is organized as follows. In section II we define the Omega network and describe some of its properties. This definition is then generalized in section III. The main problem of partitioning a set of conflicting messages is introduced in section IV. In section V we generalize a lower bound given in [1], and in the following sections we consider the complexity of the problem introduced in section IV.

II. PRELIMINARIES

In [8] Lawrie proposed the Omega network as an interconnection network for a multiprocessor system. Following Lawrie[9], an N-input N-output Omega network (also called an N × N Omega network), where $N=2^m$, consists of $\log_2 N$ identical stages. Each stage implements a *perfect shuffle* interconnection[13] followed by N/2 switching elements. In Figure 1(a) we show an 8×8 Omega network, and in Figures 1(b) and 1(c) we show the two possible states for each of the switches. Figure 1(b) shows the "straight through" state where the input signals are sent directly to the corresponding

---

[†] Research supported in part by NSF Grant MCS 83-03932.
[‡] Research supported in part by NSF Grant DCR 86-03184.

outputs, and 1(c) shows the "interchange" state where the input signals are first interchanged before being sent to the outputs. Note that in Figure 1(a) we have labeled the interconnection links for each stage with a $\log_2 N$ bit binary address. We have also numbered the stages, and shown how a path from input 100 to 110 is established through the network.

A particular path through the network can be represented by a source-destination pair, abbreviated as an (s,d) pair. The source (s) $s_0 s_1 \cdots s_{m-1}$ being the binary address of the input at the first stage and the destination (d) $d_0 d_1 \cdots d_{m-1}$ being the binary address of the output at the last stage, where $m = \log_2 N$. Careful examination of the network shows that the *path code* $s_0 s_1 \cdots s_{m-1} d_0 d_1 \cdots d_{m-1}$ completely determines a unique path through the network. Specifically, if we define an m bit window W_i as the bit pattern beginning at bit position i of the path code then we see that at stage i in the network, $0 \leq i \leq m-1$, the path which goes from $s_0 s_1 \cdots s_{m-1}$ to $d_0 d_1 \cdots d_{m-1}$ makes use of the link with address $W_i = s_i s_{i+1} \cdots s_{m-1} d_0 d_1 \cdots d_{i-1}$. For example, Figure 1(a) shows a path from 100 to 110. Note that W_1 is 001 and at stage 1 the path makes use of the link with address 001.

The fact that a path code uniquely determines a path through the network is very important because it enables conflicts in the network to be detected easily. Two paths in the network are said to *conflict* if and only if they both require use of a common link in the network. Thus two (s,d) pairs are said to conflict if and only if there exists an i such that the two (s,d) pairs have the same bit pattern on window W_i. For example, Figure 2 shows how the paths (000,100) and (100,110) are established through the network. Note that both (s,d) pairs have the same bit pattern 001 on W_1, and hence both require use of the link with address 001 at stage 1.

Another important property of Omega networks is that they are "bit controlled." Given that a path is to be established from input $s_0 s_1 \cdots s_{m-1}$ to output $d_0 d_1 \ldots d_{m-1}$, all that is required to set the state of the switch at stage i is d_i. If $d_i = 0$ then the path is routed through the upper output of the switch. If $d_i = 1$ then the path is routed through the lower output. We also note that all of these properties extend to every network in the shuffle-exchange class.

Throughout the rest of the paper the reader is encouraged to think of the network as if it operates in *circuit switching mode*. In other words, if a message is to be sent through the network then the switches in the network are set so as to establish a path between the source (input) and destination (output). Once a path has been established, it is maintained as long as data is to be transferred. Only after the transfer is complete may the path be broken. This is in contrast to *packet switching* mode, where packets of data are routed through the network without a physical path ever being established between the source and destination. Though the results in this paper hold for either model, the problems considered are more in the spirit of the circuit switching methodology.

III. A GENERALIZATION OF THE OMEGA NETWORK

In this section we generalize our notion of an Omega network by the addition of a new parameter b called the *bundling factor* of the network. Specifically, we define a $bN \times bN$ Omega network, where $N = 2^m$, to have bundling factor b if each switch in the network has two bundles of inputs and outputs each of size b. Figure 3 shows a 12×12 Omega network where $b = 3$. Notice that each input bundle may carry b or fewer signals into a switch. Hence, a total of at most 2b signals may be input to a switch at any given time. Similarly, each output bundle may carry b or fewer signals out of a switch. For each input bundle, the incoming signals may be sent to the upper or lower output bundle. But note that all the signals for a given input bundle don't necessarily have to go to the same output bundle. Some may go to the upper bundle, while others may go to the lower. If more than b require use of the same output bundle then we say that *congestion* occurs. Note that the standard definition of the Omega network given in section II is just a special case of the generalized definition where b=1. Similarly "conflict" is just a special case of "congestion". An example of such a network where b>1, is in the proposed G.E. Cross Omega machine where b=16[6]. The concept of bundling is also discussed in[14].

Recall from section II that two (s,d) pairs are said to conflict at stage i in the network if and only if they have the same bit pattern on window W_i. But note that when b>1 the fact that two pairs have the same bit pattern on window W_i doesn't necessarily imply that congestion occurs. In order for congestion to occur at stage i, at least b+1 pairs must have the same bit pattern on window W_i. For example, suppose the paths (0000,1000) and (1100,1001) are established on a 32×32 Omega network where b=2. Note that these pairs have the same bit pattern on W_2 and hence the bundle at stage 2, whose address is 0010, is full. If at this point we tried to establish the path (0100,1010), which also has the bit pattern 0010 on W_2, then congestion would occur.

IV. PARTIAL PERMUTATIONS

In this section we introduce the problem to be considered throughout the rest of the paper. Specifically, given a set S of (s,d) pairs, it may not be possible to set up all of the required paths simultaneously. This is because the set S may give rise to congestion in the network. In order to implement the communication specified by the set S, one may partition S into disjoint subsets $S_1, S_2, \ldots, S_K$ such that each subset S_i, $1 \leq i \leq K$, is congestion free. Such a scheme is said to implement the specified communication in K *rounds.* Clearly, in order to realize the communication efficiently, it is important to minimize the value of K.

Before formally defining the problem, we first define a particular type of message pattern called a *partial permutation.* Such a message pattern is of particular interest in SIMD applications where regular message patterns are likely to arise.

Definition: Given a set S of (s,d) pairs, if $s_i \neq s_j$ and $d_i \neq d_j$ for all (s_i,d_i), $(s_j,d_j) \in S$, where $i \neq j$, then we say that S forms a *partial permutation.*

Note that in the special case where every input to the network is connected to some output then the partial permutation is actually a (complete) permutation. Furthermore, observe that the size of the network on which the message pattern is to be routed is encoded in the set S since each source and destination is represented by exactly $m=\log_2 N$ bits. This is true regardless of what kind of message pattern is represented by the set S, partial permutation or otherwise. We now formally define the problem to be considered in this and the following sections, which we call **K,b-Round Routing of a Partial Permutation on an Omega Network**, abbreviated K,b-RP.

INSTANCE: Set S of (s,d) pairs representing a partial permutation to be routed on a bN×bN Omega network; integers b and K (N is implicit in S).

QUESTION: Is there a partition of the set S into K disjoint sets such that each set is congestion free when routed on a bN×bN Omega network?

This problem has been considered by other authors, but only when the message pattern is a (complete) permutation and only for Omega networks where b=1. In [1] it is shown that given a set S representing a (complete) permutation one can establish a lower bound on the value of K. In section V we restate this result, in our own terms, and then give a simple generalization of it for networks where $b>1$. We also note that the result holds for partial, as well as complete permutations. In [15] a "conflict resolution scheme" is given which will partition a set S, representing a (complete) permutation, into some number of nonconflicting rounds, however, as the authors point out, their scheme is not guaranteed to find the minimum number of rounds. In [10] a polynomial algorithm is given which will partition such a set into the minimum number of rounds, but only when the (complete) permutation is in the "bit permute complement" class (we would like to point out, however, that this class does include many frequently encountered message patterns). The authors also show that any instance of K,1-RP representing a (complete) permutation can be reduced to the Graph K-colorability problem. However, this reduction does not prove that the problem is computationally as hard as the Graph K-colorability problem. Specifically, it does not establish an NP-hardness result. In sections VI-VIII we consider the complexity of some special cases of K,b-RP, some of which we show to be solvable in polynomial time, others we show to be NP-complete. From this it follows that K,b-RP is also NP-complete.

V. A LOWER BOUND ON K

Given a set S of (s,d) pairs representing a partial permutation, one can construct a hypergraph $H=(V,E)$ as follows. Assume that p (s,d) pairs are stored in a $p \times 2m$ matrix, one pair per row, where each pair consists of 2m bits. First, for each pair $(s_i,d_i) \in S$ create a vertex v_i and add it to V. Next, imagine an m bit window sliding across the matrix from left to right (see figure 4). For each window W_i, $1 \leq i \leq m$, sort the (s,d) pairs based upon the bit pattern in W_i. This places all pairs which conflict i.e., have the same m bit pattern on W_i, on adjacent rows. From a scan of this ordered list of pairs construct each maximal set of (s,d) pairs which have the same m bit pattern on W_i. For each such maximal set, a hyperedge is placed between the corresponding vertices. Straightforward implementation of the above procedure would give an algorithm which operates in polynomial time. Note this could create two hyperedges $e_1,e_2 \in E$ such that $e_1 \subset e_2$. For example, Figure 5 shows a partial permutation consisting of three (s,d) pairs. The first two induce the hyperedge e_1 because they agree on W_1, while all three induce the hyperedge e_2 because they agree on W_2.

FACT: Given a set S representing a partial permutation and the corresponding hypergraph $H=(V,E)$ as constructed above

$$\max_i(\,|e_i|\,) \leq r, \quad e_i \in E$$

where r is the number of rounds necessary for the given partial permutation[1].

In proof, consider a hyperedge $e_i \in E$ and let $|e_i|=u$. By construction of H, there exists u (s,d) pairs which have the same bit pattern on window W_i for some i, $1 \leq i < \log_2 N$. Hence these u pairs all require use of the same link at stage i in the network. It follows that at least u rounds are required by the partial permutation. As previously stated, this result has already been established, although by means of a different model, and only as it applies to (complete) permutations. We now generalize this result to apply to Omega networks where $b>1$.

THEOREM 1: Given an instance of K,1-RP and the corresponding hypergraph $H=(V,E)$ as constructed above

$$\frac{\max_i(|e_i|)}{b} \leq r, \quad e_i \in E$$

where r is the number of rounds necessary for the partial permutation.

PROOF: see [3].

COROLLARY:

$$\frac{\max_i(|e_i|)}{r} \leq b, \quad e_i \in E$$

This could be interpreted as saying that given a partial permutation and a specified number of rounds one can determine the smallest possible bundling factor for the network on which the partial permutation is to be routed in the specified number of rounds. Note this can be computed easily in polynomial time.

VI. FIXED K=1 AND FIXED b≥1

In this and the following sections we consider some special cases of K,b-RP. The first special case we consider arises when K=1. Here we show that regardless of the value of b, this case can be solved in polynomial time. In such a case the problem reduces to verifying that at most b of the (s,d) pairs have the same bit pattern on each window W_i, $1 \leq i < \log_2 N$. If more than b have the same bit pattern on W_i, for some i, then we know that at stage i in the network more than b require use of the same bundle. Hence if all the paths were established in one round, congestion would occur.

Algorithm A.1 in the appendix of [3] will take care of this case. Intuitively, the algorithm works as follows. Assume the (s,d) pairs are stored in a matrix, and imagine an m bit window, where $m = \log_2 N$, sliding from left to right over the matrix (see Figure 4). As the window passes over window W_i, it first sorts the (s,d) pairs based upon the bit pattern in window W_i. Note this partitions the pairs into blocks such that all pairs in a given block conflict i.e., have the same bit pattern, on window W_i. The algorithm then counts the number of pairs in each block. It terminates when it either scans all the windows and finds no congestion, or when it finds that some window contains more than b pairs which have the same bit pattern on W_i. Clearly the algorithm operates in polynomial time.

VII. FIXED b=1

In this section we consider some special cases of the above problem which arise when b=1. In the simplest case b=K=1. Note that b=1 means we are dealing with the standard Omega network, while K=1 suggests that the problem is only to determine if the set S is conflict-free. From section VI we know this problem can be solved in polynomial time.

Next we consider the case where b=1 and K=2. Like the previous case, this too can be solved in polynomial time. To see this, just note that any such instance can be reduced to the Graph 2-colorability problem as follows[10]. Create a vertex for each (s,d) pair in S and place an edge between two vertices if there is a conflict between the corresponding pairs. An easy proof shows that the resulting graph is 2-colorable if and only if S can be partitioned into two conflict-free rounds. Since the Graph 2-colorability problem can be solve in polynomial time[5], so can the corresponding 2,1-RP problem.

We now show that for $b=1$ and any fixed $K \geq 3$ the problem becomes NP-complete.

THEOREM 2: K,1-RP is NP-complete for any fixed $K \geq 3$.

PROOF: (sketch)

Given an instance of K,1-RP, a nondeterministic algorithm could guess a K-partition of the (s,d) pairs and verify that no two pairs in the same set have the same bit pattern on W_i, $1 \leq i < \log_2 N$, in polynomial time. Hence the problem is in NP.

For the NP-hardness, we reduce Graph K-colorability with maximum vertex degree 2(K-1) to K,1-RP as follows (For a proof of the NP-completeness of the Graph K-colorability problem with maximum vertex degree 2(K-1) we refer the reader to Theorem A.1 in the appendix of [3]). Given an undirected graph $G=(V,E)$ we create two types of (s,d) pairs for the routing problem: primary pairs, and enforcer pairs. The primary pairs will be used to model the constraint that two adjacent vertices in G cannot be assigned the same color, hence we create two (s,d) pairs for each edge $(v_i,v_j) \in E$. The binary representations of these pairs will be created in such a way that they conflict on some window, in effect forcing the two pairs to be placed in different rounds. Thus, for each vertex $v_i \in V$ we create a total of $d(v_i)$ primary pairs (s_j^i,d_j^i), $1 \leq j \leq d(v_i)$.

As one may suspect, the K rounds in the routing problem will correspond, somewhat directly, to the K colors used to color G. Given this, we would like to force all d(v) primary pairs corresponding to a vertex v to be placed in the same round. We do this by creating, for each vertex $v_i \in V$, K-1 enforcer pairs $(s_{e_1}^i,d_{e_1}^i)$, $(s_{e_2}^i,d_{e_2}^i),\ldots,(s_{e_{K-1}}^i,d_{e_{K-1}}^i)$. These pairs are created in such a way that the following conflicts occur: 1) $(s_{e_p}^i,d_{e_p}^i)$ and $(s_{e_q}^i,d_{e_q}^i)$ conflict, for all $1 \leq p,q \leq K-1$, and 2) each primary pair (s_j^i,d_j^i), $1 \leq j \leq d(v_i)$, conflicts with each enforcer pair $(s_{e_p}^i,d_{e_p}^i)$, $1 \leq p \leq K-1$.

Note that if only the conflicts mentioned above occur then each enforcer for a given vertex must be placed in a different round. Since each primary pair (s_j^i,d_j^i), $1 \leq j \leq d(v_i)$, conflicts with each enforcer for v_i, all primary pairs for v_i are forced to be placed in the same round. Since there are no conflicts between the primary pairs for a given vertex, we know that no conflicts will exist between the primary pairs for v_i within this round. The fact that all primary pairs for a given vertex are forced to be placed in the same round models the constraint that a given vertex can only be assigned a single color.

Recalling now that for each edge $(v_i,v_j) \in E$ some primary pair (s_k^i,d_k^i), $1 \leq k \leq d(v_i)$, conflicts with some other primary pair (s_m^j,d_m^j), $1 \leq m \leq d(v_j)$, we conclude the following: the paths represented by the (s,d) pairs resulting from the above construction can be established on an N×N Omega network in K non-conflicting rounds if and only if G is K-colorable.

Assume that G is K-colorable, let the coloring function be denoted by *f*, and assign the (s,d) pairs to rounds as follows. For each $v_i \in V$ assign the primary pairs (s_j^i,d_j^i), $1 \leq j \leq d(v_i)$, to round $f(v_i)$, and the enforcer pairs to the other K-1 rounds arbitrarily, one per round. We claim that if the pairs are constructed in such a way that only the conflicts mentioned above occur, then this assignment of pairs to rounds will result in K conflict-free rounds. For suppose not, and consider pairs (s,d) and (s',d') which conflict, and assume without loss of generality that they were placed in round 1. Case 1) (s,d) is an enforcer pair for some vertex v. By construction of the pairs, the only other pairs with which (s,d) conflicts are the primary pairs for vertex v and the other enforcer pairs created for v. But by the above assignment of pairs to rounds, the primary pairs for v were placed in one round and the enforcer pairs were placed in the other K-1 rounds arbitrarily, one per round. Thus, assuming only those conflicts mentioned above occur, no enforcer pair can occur in a conflict. Case 2) (s,d) is a primary pair. By construction of the pairs, the only other pairs with which (s,d) conflicts are the enforcer pairs for v and one other primary pair for some other vertex v' which is adjacent to v. From case 1) we know that (s',d') could not be one of the enforcers for v. Hence (s',d') must be a primary pair created for some vertex v' which is adjacent to v. But since v and v' are adjacent they must be assigned different colors and hence, by the above assignment of pairs to rounds, their primary pairs must be in separate rounds. Thus no primary pair can occur in a conflict.

Now assume the (s,d) pairs can be partitioned into K conflict-free rounds, and consider the $d(v_i)+K-1$ pairs created for a vertex $v_i \in V$. By construction of the pairs we know the K-1 enforcers conflict with each other and hence must have been placed in different rounds. Assume they were placed in rounds $\alpha_1, \alpha_2, \ldots, \alpha_{K-1}$ where $1 \leq \alpha_i \leq K$ and $\alpha_i \neq \alpha_j$, $1 \leq i,j \leq K-1$. Furthermore, all primary pairs (s_j^i, d_j^i), $1 \leq j \leq d(v_i)$, conflict with each enforcer, and since none of these primary pairs conflict among themselves, they must all be in round γ, where $1 \leq \gamma \leq K$, $\gamma \neq \alpha_i$, for all $1 \leq i \leq K-1$. Thus we assign $f(v_i)=\gamma$. Hence the color of any vertex is determined by the round which the corresponding primary pairs are placed in. Given this assignment of colors to vertices we know that no two adjacent vertices $v_i, v_j \in V$, $1 \leq i,j \leq |V|$, are assigned the same color, otherwise their primary pairs would have been placed in the same round which we know can't happen since some primary pair for v_i and some primary pair for v_j conflict. Hence the above assignment does induce a K-coloring of G. □

For a description of the construction of the (s,d) pairs so as to induce the conflicts mentioned above see [3].

VIII. FIXED $b \geq 2$

In this section we consider some special cases of K,b-RP which arise when $b \geq 2$. In the simplest case $b=2$ and $K=1$. From section VI, we know that this case can be solved in polynomial time. Next, we consider the case where $b \geq 2$ and $K \geq 2$ and show that the problem becomes NP-hard.

THEOREM 3: K,b-RP is NP-complete for any fixed $K \geq 2$, and any fixed $b \geq 2$.

PROOF: see [3].

IX. CONCLUSIONS

In this paper we have considered the complexity of some problems which arise in the use of Omega networks. Specifically, we have considered the problem of partitioning a set of conflicting messages into a minimum number of nonconflicting subsets. Though we have shown this problem to be NP-complete for a number of cases, we do not suggest that the strategy of partitioning a set of messages into a minimum number of rounds is entirely impractical. Which problems are being solved, and ultimately what message patterns develop will determine whether or not this strategy should be used. An example of a particular class of message patterns for which this problem can be solved in polynomial time is given in [10]. For an example of a strategy to deal with conflicts in a single stage *packet switching* network see[7].

Finally, we point out that throughout the literature, numerous networks have been shown to be topologically equivalent to the Omega network[1],[15]. These networks have sometimes been collectively referred to as the *shuffle exchange* class[15]. The (s,d) pairs created in any of our proofs could have been addresses for the appropriate inputs and outputs of another network in the shuffle-exchange class. Hence our results extend to these networks as well.

References

[1] Agrawal, D.P., "Graph Theoretical Analysis and Design of Multistate Interconnection Networks," IEEE Trans. Comput., vol. C-32, no. 7, July 1983, pp. 637-648.

[2] Benes, V.E., "On Rearrangeable Three-Stage Connecting Networks," B.S.T.J., 41, 1962, pp.1481-1492.

[3] Bernhard, P.J. and Rosenkrantz, D.J., "The Complexity of Routing Through an Omega Network," Comput. Sci. Dept., SUNY Albany, Albany, N.Y., TR 87-12, 1987.

[4] Feng, T, "A Survey of Interconnection Networks," IEEE Comput., vol. 14, no. 12, Dec. 1981, pp. 12-27.

[5] Garey,M.R. and Johnson,D.S., "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H.Freeman and Co., 1979.

[6] Hardy,R.M., personal communication.

[7] Lang, T., "Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network," IEEE Trans. Comput., vol. c-25, no. 5, May 1976, pp.496-503.

[8] Lawrie,D.H., "Memory-processor connection networks," Univ. Illinois, Urbana-Champaign, Dep. Comput. Sci., Rep. 557, Feb. 1973.

[9] Lawrie,D.H., "Access and Alignment of Data in an Array Processor," IEEE Trans. Comput., vol. C-21, no. 12, Dec. 1975, pp. 1145-1155.

[10] Raghavendra,C.S. and Varma,A., "Fault-Tolerant Multiprocessors with Redundant-Path Interconnection Networks," IEEE Trans. Comput., Vol. C-35, No. 4, April 1986, pp. 307-316.

[11] Siegel, H.J., "Interconnection Networks for SIMD Machines," Computer, vol. 12, No. 6, June 1979, pp.57-66.

[12] Siegel, H.J., "Interconnection Networks for Large-Scale Parallel Processing," Lexington Books, 1985.

[13] Stone,H.S., "Parallel processing with the perfect shuffle," IEEE Trans. Comput., vol. C-20, Feb. 1971, pp. 153-161.

[14] Szymanski, T.H. and Hamacher, V.C., "On the Permutation Capability of Multistage Interconnection Networks," IEEE Trans. Comput., vol. C-36, no. 7, July 1987, pp. 810-822.

[15] Wu, C.L., Feng, T.Y., "On a Class of Multistage Interconnection Networks," IEEE Trans. Comput., vol. C-29, no. 8, August 1980, pp.694-702.

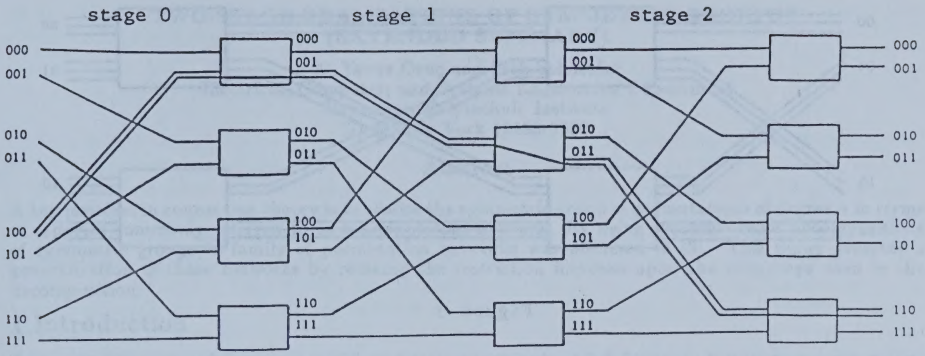

stage 0
stage 1
stage 2
000
001
010
011
100
101
110
111

Figure 1(a)

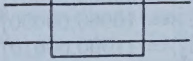


Figure 1(b)

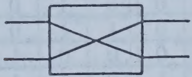


Figure 1(c)

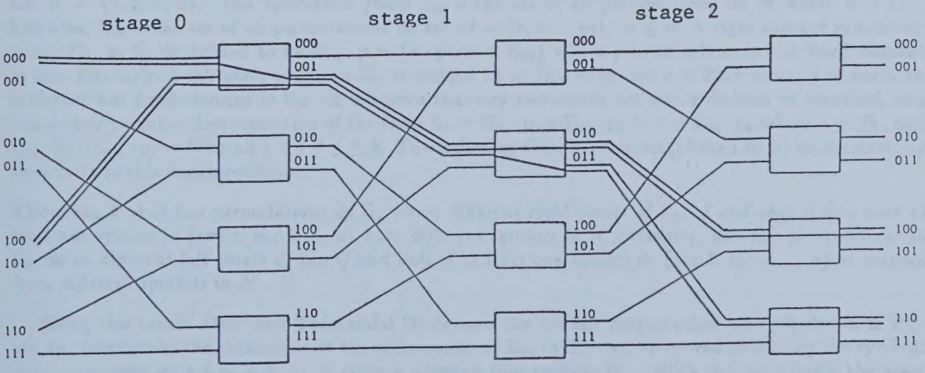

stage 0
stage 1
stage 2
000
001
010
011
100
101
110
111

Figure 2

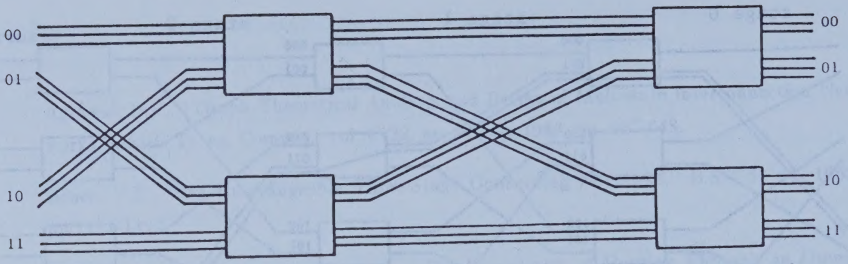
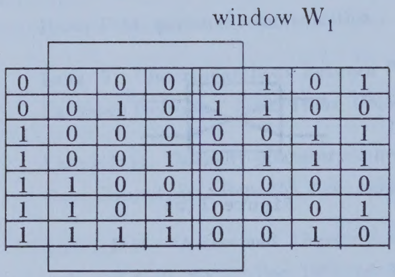

00
01
10
11
00
01
10
11

Figure 3



0	0	0	0	0	1	0	1
0	0	1	0	1	1	0	0
1	0	0	0	0	0	1	1
1	0	0	1	0	1	1	1
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	0

Figure 4

$$(s_1,d_1)=(00000,00001)$$
$$(s_2,d_2)=(10000,00000)$$
$$(s_3,d_3)=(11000,00010)$$

$$e_1=\{v_1,v_2\}$$
$$e_2=\{v_1,v_2,v_3\}$$

Figure 5

TWO-STAGE REALIZATIONS OF SYMMETRIC GROUPS* (EXTENDED SUMMARY)

A. Yavuz Oruç and Seth Schneider
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Abstract

A key problem in connection theory is to realize the symmetric group of permutations of degree n in terms of a device commonly referred to as a permutation network. By using recursive coset decompositions of symmetric groups, a family of permutation networks was obtained in [3]. This paper presents a generalization of these networks by relaxing the restriction imposed upon the subgroups used in the decomposition.

1 Introduction

A permutation network is a device with two sets of terminals called *inputs* and *outputs* and can realize all one-to-one mappings from its inputs onto its outputs. A single stage permutation network is a complete bipartite graph with n^2 edges for n inputs and n outputs. The bipartite graph realization has the drawback of $O(n^2)$ edges and $O(n)$ fan-in and fan-out, though it has a trivial set-up procedure. As an alternative Oruc and Thirumalai [3] examined 2-stage permutation networks and their recursive decompositions, and showed that such networks are firmly tied with right and left coset decompositions of the symmetric groups of permutations on n letters. They presented a design space to customize the fan-in/fan-out and number of edges in the target permutation network. The networks in this design space were shown to have a fan-in/fan-out ranging from 2 to n and $O(n^2)$ edges. The 2-stage network described in Oruc and Thirumalai realizes the symmetric group of degree n by assuming the availability of a network which can realize the symmetric group of degree m where $m \leq n$ and a network which is referred to as a *coset generator.* The first network is recursively decomposed to generate the networks in the design space. This paper extends this construction by replacing the first stage by a collection of permutation networks, each acting on a distinct set of symbols. Thus, rather than feeding all m inputs into the same permutation network, we partition them into a collection of sets and feed each into a separate permutation network. This extension provides a second degree of freedom and enhances the original design space reported in Oruc and Thirumalai. As a result, it is shown that a network with fewer edges, albeit with the same order of magnitude, can be constructed.

2 Notation and Background

Let $N = \{1, 2, ..., n\}$. The *symmetric group* Σ_n is the set of all permutations on N where $n = |N|$. Likewise, Σ_m is the set of all permutations on set $M = \{1, 2, ..., m\}$; $m \leq n$. A *right coset* of symmetric group Σ_m in Σ_n is defined to be $\Sigma_m \cdot p = \{\sigma \cdot p : \sigma \in \Sigma_m\}$ where p is an arbitrary but fixed element in Σ_n. Similarly, a *left coset* of Σ_m in Σ_n is defined as $p \cdot \Sigma_m = \{p \cdot \sigma : \sigma \in \Sigma_m\}$ where p is again an arbitrary but fixed element in Σ_n. It is known that any two cosets are either disjoint or identical, and hence they permit a decomposition of the form $\Sigma_n = \Sigma_m \cdot p_1 + \Sigma_m \cdot p_2 + \cdots + \Sigma_m \cdot p_k$ where $k = \frac{n!}{m!}$, and $\Sigma_m \cdot p_i \cap \Sigma_m \cdot p_j = \emptyset$ for all $1 \leq i \neq j \leq k$. The following theorem was established in [1] to characterize the cosets in this decomposition.

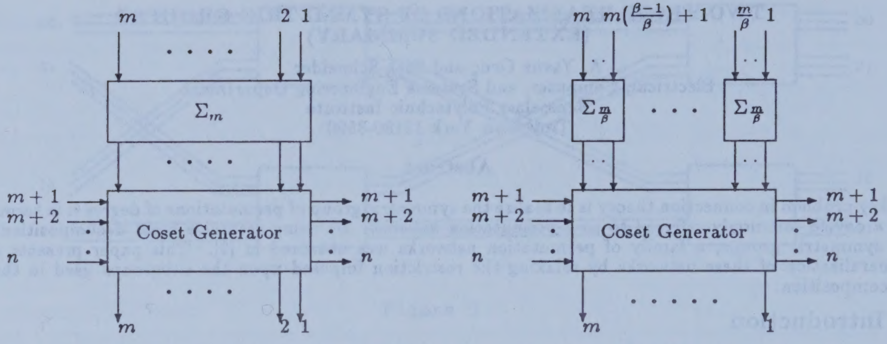
Theorem 1 *Any two permutations in Σ_n lie in different right cosets of Σ_m if and only if they map at least one symbol in $\{m+1, m+2, ..., n\}$ onto different symbols in N. Similarly, any two permutations in Σ_n lie in different left cosets of Σ_m if and only if at least one symbol in $\{m+1, m+2, ..., n\}$ is mapped from different symbols in N.* ||

Using this result, Oruc and Thirumalai [3] devised the 2-stage permutation network shown in Figure 1a. Intuitively, the realization of the right cosets of Σ_m in Σ_n can be viewed as mixing the symbols $1, 2, \ldots, m$ with $m+1, m+2, \ldots, n$ given a network that realizes Σ_m. With this as a basis, the coset generator portion of the network in Figure 1a was specified to have $(n-m)n + m(n-m+1)$ edges. The first term in the expression accounts for the number of edges between the horizontal inputs and all outputs, while the second one corresponds to the number of edges between the vertical inputs and all outputs. It was also shown that, when the top subnetwork is recursively decomposed, the total number of edges is given by the formula

$$C_n = n^2 - \frac{n}{2} + \frac{n^2}{2(n-m)}. \tag{1}$$

---

*This work was supported in part by the National Science Foundation under grant CCR-8708864.



1a : Network with a single subgroup   *1b : Network with β subgroups*

Figure 1: 2-stage networks based on coset generators.

This formula accounts for the edge complexity of all permutation networks which can be derived from Theorem 1. By varying m, various permutation networks ranging from cellular permutation arrays [1] to crossbars may be obtained.

3 Extended Topology

The network described above is a special case of a more general coset decomposition of Σ_n. In this section we describe the network realization of this decomposition and provide a formula for its edge complexity. We first note that Theorem 1 assumes that the subgroup with respect to which the cosets are characterized is fixed to Σ_m. The theorem was generalized in [2] as follows. Let $M_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}\}; 1 \leq i \leq \beta$ such that $M_1 + M_2 + \ldots + M_\beta = N$, and let Σ_{M_i} be the symmetric group over M_i. Define

$$p = \begin{pmatrix} a_{1,1}a_{1,2}\cdots a_{1,m_1} & a_{2,1}a_{2,2}\cdots a_{2,m_2} & \cdots & a_{\beta,1}a_{\beta,2}\cdots a_{\beta,m_\beta} \\ x_{1,1}x_{1,2}\cdots x_{1,m_1} & x_{2,1}x_{2,2}\cdots x_{2,m_2} & \cdots & x_{\beta,1}x_{\beta,2}\cdots x_{\beta,m_\beta} \end{pmatrix}$$

$$q = \begin{pmatrix} a_{1,1}a_{1,2}\cdots a_{1,m_1} & a_{2,1}a_{2,2}\cdots a_{2,m_2} & \cdots & a_{\beta,1}a_{\beta,2}\cdots a_{\beta,m_\beta} \\ y_{1,1}y_{1,2}\cdots y_{1,m_1} & y_{2,1}y_{2,2}\cdots y_{2,m_2} & \cdots & y_{\beta,1}y_{\beta,2}\cdots y_{\beta,m_\beta} \end{pmatrix}.$$

Theorem 2 *The right cosets* $(\Sigma_{M_1}\Sigma_{M_2}\cdots\Sigma_{M_\beta})\cdot p$ *and* $(\Sigma_{M_1}\Sigma_{M_2}\cdots\Sigma_{M_\beta})\cdot q$ *are disjoint if and only if there exists an index* $i; 1 \leq i \leq \beta$ *such that* $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}\} \neq \{y_{i,1}, y_{i,2}, \ldots, y_{i,m_i}\}$.

Similarly, the left cosets $p^{-1}\cdot(\Sigma_{M_1}\Sigma_{M_2}\cdots\Sigma_{M_\beta})$ *and* $q^{-1}\cdot(\Sigma_{M_1}\Sigma_{M_2}\cdots\Sigma_{M_\beta})$ *are disjoint if and only if there exists an index* $i; 1 \leq i \leq \beta$ *such that* $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}\} \neq \{y_{i,1}, y_{i,2}, \ldots, y_{i,m_i}\}$.

This theorem is stated for $N = M_1 + M_2 + \ldots + M_\beta$. In order to remain consistent with the realization given in the preceding section we shall assume that $M_1 + M_2 + \ldots + M_\beta = \{1, 2, \ldots, m\}$, and that $\{1, 2, \ldots, m\} \subset N$. The symbols $m+1, m+2, \ldots, n$ will then denote the horizontal inputs to the coset generator to be constructed. We also assume that $m = m_1 = m_2 = \ldots m_\beta$.

With the above assumptions, Theorem 2 induces the 2-stage network shown in Figure 1b. The top stage consists of β subnetworks, each with $\frac{m}{\beta}$ inputs and outputs. The edges inside the coset generator at the bottom can be specified as follows. Choose m outputs and divide them into groups of $\frac{m}{\beta}$. Next connect the m vertical inputs to the chosen m outputs in a one-to-one, onto fashion. Now connect the first $\frac{m}{\beta}$ inputs, via a complete bipartite graph, to the union of all the outputs not of the originally chosen m, and all but the first $\frac{m}{\beta}$ outputs. Repeat these connections for all β subgroups in a similar fashion. Finally, connect the $(n-m)$ horizontal inputs to all n outputs. It can be established that the edges specified are both necessary and sufficient to realize Σ_n by the 2-stage network shown in Figure 1b.

With the edges as specified above, we can evaluate the number of edges in the coset generator to be

$$n^2 + m - \frac{m^2}{\beta}, \tag{2}$$

and letting C_n represent the total number of edges in the network, the recurrent expression is

$$C_n = n^2 + m - \frac{m^2}{\beta} + \beta C_{\frac{m}{\beta}} \tag{3}$$

By recursively decomposing the subnetworks at the top until they can no longer be decomposed, we obtain the final cost expression to be

$$C_n = n^2 - \frac{\beta}{\beta-1}n + \frac{\beta n - m}{\beta - 1} log_\beta \frac{\beta n - m}{n - m}. \tag{4}$$

We now have two parameters, namely m, which dictates the number of vertical and horizontal inputs, and β, which dictates the number of subnetworks the top network is divided into. If we decrease m, that is, increase the number of horizontal inputs, we minimize the edge complexity or cost of the network. Also, by increasing β the total complexity decreases. The extreme case would then be when $m = 0$, creating n horizontal inputs or $\beta = m$, creating m top subnetworks consisting of one input each. In both of these cases, the resulting overall network would be that of a single stage, $n - input$ crossbar.

Although the edge complexity only approaches $O(n^2)$, that is, it is strictly greater than or equal to that of a crossbar, a crossbar has the drawback of a large fan-out of its inputs. A crossbar of n symbols has a fan-out of n for each of its inputs while the topology presented in this section has a fan-out directly related to the two parameters, m and β. More precisely, it can be shown that the vertical m inputs of the coset generator have a fan-out of $n - m + \frac{m}{\beta}(\beta - 1) + 1$ and the horizontal $(n - m)$ inputs have a fan-out of n, both of which are less than or equal to that of a crossbar. Although the horizontal inputs have a constant fan-out of n, the number of horizontal inputs may be varied, leading to fewer inputs with a fan-out of n. A similar argument may be applied to the fan-in of the outputs of the coset generator versus those of a crossbar.

4 Set-up Procedure for Extended Network

The set-up procedure described in Oruc and Thirumalai can be extended for the networks described in the preceding section. Basically, given a permutation, the problem is to determine which edges to activate inside the coset generator and the submaps for the networks in the first stage. To carry out these steps, we note that the permutation of the network depicted in Figure 1b can be expressed as $\Sigma_n = (\Sigma_{M_1} \cdot \Sigma_{M_2} \cdots \Sigma_{m_\beta}) \cdot R$ where Σ_{M_i} is the symmetric group over M_i and R is a set of distinct coset leaders of $\Sigma_{M_1} \cdot \Sigma_{M_2} \cdots \Sigma_{M_\beta}$ in Σ_n. Consequently, each $p \in \Sigma_n$ can be expressed as $p = \sigma_1 \cdot \sigma_2 \cdots \sigma_\beta \cdot r$ where $\sigma_i \in \Sigma_{M_i}$ and $r \in R$. Thus, setting up the network for p reduces to determining σ_i; $1 \leq i \leq \beta$ and r given p.

Let $M = M_1 + M_2 + \cdots + M_\beta$ and suppose that the inputs in $N - M$ are mapped to outputs $y_1 y_2 ... y_{n-m}$. This is easily established by marking the appropriate edges in the coset generator. As for the inputs that belong to M_i; we consider these in two groups. Those that are to be connected to the outputs which are labelled by symbols in M_i and those which are not. The submap between the first group of inputs and corresponding outputs is realized by marking the edges which are connected to those outputs inside the coset generator and by accordingly rearranging the subnetwork with inputs M_i. The submap between the second group of inputs and corresponding outputs are then easily realized since a complete bipartite graph is embedded between the inputs in M_i and all outputs in $N - M_i$. This completes the set-up procedure. It can further be shown that this procedure takes $O(n)$ steps. The same procedure can be used recursively to set up the networks described in Section 3. In this case, the set-up time becomes $O(kn)$ where k denotes the depth of the recursion.

5 Concluding Remarks

This paper extended the results presented in Oruc and Thirumalai [3]. It has shown how to design a permutation network based upon coset decomposition of symmetric groups into several smaller subgroups. An interesting implication of this procedure is that the network size can be predetermined by the network designer by selecting two independent parameters. As a result, both stage count and edge count can be minimized concurrently using two unrelated parameters.

REFERENCES

1. A. Y. Oruc, "Designing cellular permutation networks through coset decomposition of symmetric groups," *Journal of Parallel and Distributed Computing*, Vol. 3, No. 3, Aug. 1987, pp. 402-422.

2. A. Y. Oruc and M. Y. Oruc, "Permuting by decomposing symmetric groups into cosets," In review.

3. A. Y. Oruc and A. Thirumalai, "New cellular permutation networks for parallel communications," *In Proc. of Information and Systems Sciences Conference, 1987, Johns Hopkins Univ.*,pp. 465-471.

Test Generation for C-testable Array Multipliers

Sin-Min Chang and Chin-Long Wey

Department of Electrical Engineering
Michigan State University, East Lansing, MI 48824

Abstract

Designs of C-testable array multipliers such as Carry-Propagate array multipliers, Carry-Save array multipliers, and Baugh-Wooley array multipliers, are presented. The test sets for these array multipliers are generated based on the graph labeling scheme proposed by Chatterjee and Abraham. It requires only 16 test patterns for testing such array multipliers irrespective of the circuit size.

I. Introduction

Rapid advances in semiconductor fabrication technology have made possible the implementation of digital circuits with a very large number of devices on a single chip. This complexity is coupled with an increase in the ratio of logic to pins which drastically reduces the controllability and observability of the logic on the chip [1]. Testing is expected to be more difficult with the higher-complexity chips. One of the important issues associated with circuit testing is fault detection. Fault detection is carried out by applying a sequence of test inputs and observing the resulting outputs. Therefore, the cost of testing includes the generation of test sequences and their application. One of the main objectives in testing is to minimize the length of the test sequence [2].

Recently, an array multiplier of reasonable size has been implemented on a single very large scale integrated (VLSI) circuit chip [3,4]. Due to the regular and iterative structure, the designs of iterative logic arrays (ILAs) that are tested with a test set of constant length irrespective of the circuit size (C-testability), has been presented [5-7]. The concept of C-testability has also been extended and slightly modified the conventional design of array multipliers to produce a C-testable design [8].

In the design methodology of [8], the fault model assumes: (1) in an array, at most one basic cell is faulty at a time; (2) the fault is permanent fault; and (3) the fault may alter the cell's output functions in any arbitrary way, as long as the faulty cell remains in combinational circuit. With the above assumptions, Shen and Ferguson have claimed that the testing of an array multiplier must involve the exhaustive testing of every cell by applying all possible input patterns and observing all output patterns. In other words, for each cell, consists of an AND gate and a full adder, of a testable array multiplier, all possible (2^4, or 16) cell input patterns must be applied. However, due to the difficulty of applying all possible input patterns, the basic cell must be slightly modified as discussed in [8]. The significance of the design methodology of [8] is that they took the advantage of the iterative structure in an array multiplier, the test sequence generated for exhaustively testing a cell can be applied to exhaustively test entire array. Consequently, the test length can be substantially reduced, and all cells can be simultaneously tested because of the repetitive nature of hardware. However, the only drawback in [8] is no systematic test pattern generation procedure was provided.

This research was supported in part by the National Science Foundation under grant MIP-8700880.

In order to systematically generate the test sequence for C-testable iterative array structures, Chatterjee and Abraham [9] have proposed a test generation methodology using a graph labeling scheme. A data-flow graph based model is formulated in which labels representing binary vectors are assigned to the branches of the data-flow graph. The labels are shown to satisfy a set of constraints imposed by cell function and interconnection topology. As a result, complex test generation problems can be solved by manipulating a set of symbolic labels with ease and efficiency [9]. The graph labeling scheme has been applied to various designs of C-testable arithmetic units, such as carry-propagate array multipliers (CPM). A design of C-testable CPM is presented in which extra XOR gates are added to produce C-testability. The proposed C-testable CPM can be tested with a test length of 8 if faults, referred to as **decoupled faults**, in the AND gates are decoupled from those in the corresponding full adders; and with a test length of 16 if faults, referred to as **coupled faults**, in the AND gates are not decoupled from those in the corresponding full adders [9]. However, this design is not free of penalty, the extra hardware has significantly degraded the speed performance.

II. Graph Labeling

Consider two sets of vectors illustrated in Table 1 [9].

Table 1. Labels.

V_1	V_2	V_3	V_4	C_1	C_2	C_3	C_4
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1

The vectors V_1, V_2, and V_3 contain all 2^3 possible binary 3-bit values. V_4 is the bitwise sum of V_1, V_2, and V_3 over GF(2). Two functions are defined [9], $g(V_i,V_j,V_k) = V_i \oplus V_j \oplus V_k$, and $f(V_i,V_j,V_k) = V_iV_j + V_iV_k + V_jV_k$. The function g is the bitwise summation, over GF(2), of the vectors V_i, V_j, and V_k, while the function f is the bitwise carry produced in the above summation. The elements C_i's in Table 1b are computed by evaluating $f(V_i,V_j,V_k)$ bitwise, where $i \neq j \neq k \neq m$, and $1 \leq i,j,k,m \leq 4$.

Let A, B, C, X, and Y be vectors that represent any of 8 vectors V_1-V_4, and C_1-C_4. A mapping VM is defined as VM(A,B,C)=XY if X=g(A,B,C) and Y=f(A,B,C). It is represented simply as ABC-->XY.

The vectors V and C are treated as labels. A data flow graph representation of the circuit is used. Each node of this graph represents a circuit module and interconnection between the modules are represented by directed arcs between the above nodes. The objective is to label all the branches of the data flow graph such that [9]:

1) Each branch is assigned a unique label.
2) The labels on the input and output branches of a graph node are consistent with the corresponding cell function.

More specifically, consider the data flow of a 4 x 4 carry-propagate array multiplier (CPM) as shown in Figure 1 [9]. Each cell has 3 inputs, x, y, and z, and 2 outputs, sum bit u and carry bit v, where $u = x \oplus y \oplus z$ and $v = xy + xz + yz$. The input x is the AND a_ib_j, where a_i and b_j are the multiplier and multiplicand bits, respectively.

In [9], a label L consisting of two sets of vectors $L_1 = (V_1, C_1, V_4)$ and $L_2 = (V_4, C_4, V_1)$ is applied to CPM, as shown in Figure 2a. The mapping,

$V_1C_1V_4$-->V_4C_4, describes that a carry vector V_4 is propagated from the rightmost cell to the leftmost cell of the first row. In order to reproduce the label L_1 in the third row, an appropriate label L_2 is chosen for the second row such that (1) the carry assigned in the rightmost cell of the second row can be propagated to the leftmost cell; and (2) the labels L_1 and L_2 are periodically reproduced in every other row. This is referred to as **two-row periodic propagation (TRPP)**.

It should be noted that the labels on the input and output branches of a graph node must be consistent with the corresponding cell function. As a result, the carry vector of the leftmost cell in the first row must be consistent with the required vector in the y-direction input of the leftmost cell in the second row. Furthermore, the sum vector of each cell in the first row is fed to the z-direction input of each cell in the second row. In other words, the first label must be chosen as a combination of vectors that can produce the same carry and sum vectors. This results that no such label can be produced.

Although the selection of the above mentioned labels L_1 and L_2 does not satisfy the constraints, with an additional XOR gate the labels can be applied to cells perfectly. More specifically, the leftmost cell in the first row produces the carry vector V_4 and the sum vector C_4. The carry vector is supposed to be fed to the y-direction input of the leftmost cell in the second row, where an input vector C_4 is expected. Although vectors V_4 and C_4 are totally different, fortunately, C_4 is bitwise complement of V_4 except when (V_1,V_2,V_3)=(000) and (111). Therefore, an additional two-inputs XOR gate can be used to make the applied label consistent, as shown in Figure 2, where an extra control signal TEST is needed. The signal TEST is set to logical 0 during the normal operation, and is set to logical 1 to complement the carry output vector during the test mode.

The use of label L, however, is not enough to apply all possible input patterns. As the application of L to array illustrated in Figure 2(a), all input patterns are applied except (001) and (110) in Vector 1. Therefore, the other label L' consisting of $L_1'=(V_1,C_4,V_4)$ and $L_2'=(V_4,C_1,V_1)$ is applied. This label is applied in the same manner as L except that the labels L_1' and L_2' are periodically reproduced in every other diagonal column, as shown in Figure 2(b) [9]. This is referred to as **two-column periodic propagation (TCPP)**.

With the application of such labels, the modified carry-propagate array multiplier has been shown to be C-testable with a test length of 8 if decoupled faults occurs; and with a test length of 16 if coupled faults occur [9].

Although the modified CPM proposed in [9] provides C-testability that can significantly reduce the time and cost for testing it, it is not free of penalty. The extra XOR gates may not cost too much in the hardware itself with today's VLSI design technology, however, the performance degradation may entirely destroy the significance of C-testable design. More specifically, for an n x n CPM, it is requires to add (n-1) XOR gates in the C-testable design of [9]. Unfortunately, these XOR gates are located at the critical path. In other words, it is required to have additional delay time for (n-1) XOR gates. In fact, the delay time of a XOR gate is virtually the same as that of a full adder, i.e., the performance of C-testable CPM proposed in [9] may be degraded about 50% of the original design.

III. Design of a C-testable Carry Propagate Array Multiplier

The inherent drawback of the C-testable CPM proposed in [9] is not its extra hardware cost for the additional XOR gates, but the performance degradation due to the extra XOR gates. In this section, an alternative design of C-testable CPM without degrading performance is presented.

3.1. Graph Labeling.

Consider the corresponding mappings,

$$V_iV_jV_k \dashrightarrow C_mV_m, \text{ or} \quad (1)$$
$$C_iC_jC_k \dashrightarrow V_mC_m, \quad (2)$$

where $i \neq j \neq k \neq m$, and $1 \leq i,j,k,m \leq 4$. Consider also the applications of the labels that are propagated in the way of combining TRPP and TCPP, i.e., two vectors are periodically propagated in one direction while the other two are in the other direction. More specifically, let M_i's, $i=1,..,4$, represent four basic cells of a CPM, as shown in Figure 3, each cell is labeled by $L_i=(L_{i1},L_{i2},L_{i3})$. The objective is to generate these four labels propagating to the entire array cyclically. In other words, according to the interconnection topology of these cells and the mappings (1) and (2), we solve for L_{ij}'s, $i,j=1,..,4$, in the following mappings,

$$L_{11}L_{12}L_{13} \dashrightarrow L_{23}L_{42} \quad (3)$$
$$L_{21}L_{22}L_{23} \dashrightarrow L_{13}L_{32} \quad (4)$$
$$L_{31}L_{32}L_{33} \dashrightarrow L_{43}L_{22} \quad (5)$$
$$L_{41}L_{42}L_{43} \dashrightarrow L_{33}L_{12} \quad (6)$$

In order to simplify and clarify our discussion, without loss of generality, we assume that $L_1=(L_{11},L_{12},L_{13})=(V_1,V_2,V_3)$. Due to the limitation of space, the proofs of the following Theorems are omitted.

Theorem 1.

The following set of labels is a solution of the mapping (3)-(6),

$$L_1=(L_{11},L_{12},L_{13})=(V_1,V_2,V_3)$$
$$L_2=(L_{21},L_{22},L_{23})=(C_2,C_1,C_4)$$
$$L_3=(L_{31},L_{32},L_{33})=(C_4,C_3,C_2) \quad (7)$$
$$L_4=(L_{41},L_{42},L_{43})=(V_3,V_4,V_1)$$

Corollary 1.

Given a label L_1 which contains either all V_i, or all C_i, the rest of labels can be generated in the same manner as discussed in Theorem 1.

Corollary 2.

There exist 24 possible sets of such labels.

Figure 4(a) illustrates the applications of labels shown in equation (9) to arrays. The labels are perfectly applied and periodically propagated through the cells in the entire array except those in the carry propagation stage, i.e., the leftmost cells of each row which are referred to as **boundary-cells**. However, the labels in the left-boundary cells still retain the same as it is supposed to be, but in different order. For example, the label of the leftmost cell in the second row of Figure 4(a) was supposed to be $(x,y,z)=(V_3,V_4,V_1)$ and now is changed to (V_4,V_3,V_1). Similarly, in leftmost cell of the third row, the label (C_2,C_1,C_4) is changed to (C_1,C_2,C_4).

The problem arises is whether or not the application of such labels to array meets the constraints, referred to as **external constraints**, imposed by cell function and interconnection topology.

As the interconnection topology shown in Figure 1, the x-direction input of each cell is the output of a 2-input AND gate. The inputs, a_i and b_j of the AND gate, must meet the following constraints: all cells in the same diagonal column are required to apply the same a_i, and all cells in the same row are needed to apply the same b_j. In addition, since the labels in equation (7) are applied with a period of two rows and two columns. Therefore, both a_i and b_j are determined as follows: both vectors 1 and 3 have the same value of a_i, so do vectors 2 and 4; and both vectors 1 and 2 have the same value of b_j, so do vectors 3 and 4. More specifically, (a_i,b_j)-(11) if the output of AND gate is a logical 1; otherwise (a_i,b_j)-(00), (01), or (10) depending upon the external constraints. Table 4 describes the application of the labels of equation (7) to array.

Table 2. Application of Labels L_i's to Array.

Vector 1				Vector 2				Vector 3				Vector 4			
a_ib_j	V_1	V_2	V_3	a_ib_j	C_2	C_1	C_4	a_ib_j	C_4	C_3	C_2	a_ib_j	V_3	V_4	V_1
*	0	0	0	*	0	0	0	*	0	0	0	*	0	0	0
0 1	0	0	1	1 1	1	1	0	0 1	0	0	1	1 1	1	1	0
*	0	1	0	*	0	1	0	*	0	1	0	*	0	1	0
1 0	0	1	1	1 0	0	1	1	1 1	1	0	0	1 1	1	0	0
1 1	1	0	0	1 1	1	0	0	1 0	0	1	1	1 0	0	1	1
1 1	1	0	1	1 1	1	0	1	1 1	1	0	1	1 1	1	0	1
1 1	1	1	0	0 1	0	0	1	1 1	1	1	0	0 1	0	0	1
1 1	1	1	1	1 1	1	1	1	1 1	1	1	1	1 1	1	1	1

Remark: "*" denotes that a_ib_i can be either 00, 01,or 10.

Table 2 shows that both a_i and b_j can be applied consistently to meet the external constraints for all cells in the array except the left-boundary cells.

3.2. Test Pattern Generation.

Consider a 4-by-4 modified carry propagate array multiplier (MCPM), as shown in Figure 4(b). The array receives two 4-bit operands and produces an 8-bit product. The c and d inputs of the cells in the rightmost diagonal column and the top row, as well as the control signals, are connected to logical 0 during multiplication. However, it is assumed that during testing these inputs are available as primary inputs to the array. Two control signals TEST1 and TEST2, as shown in Figure 4(b) are connected to the even rows and the odd rows, respectively.

Consider the case that coupled faults occur in the cells of Figure 4(b), except the left-boundary cell. It is required to apply four inputs, a_i, b_j, y, and z, to these cells. Table 3 describes the input combinations derived from Table 2.

Table 3. Input Combinations for MCPM.

Test#	Vector 1	Vector 2	Vector 3	Vector 4
1.	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
2.	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0
3.	1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0
4.	0 1 0 1	1 1 1 0	0 1 0 1	1 1 1 0
5.	0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0
6.	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0
7.	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0
8.	1 0 1 1	1 0 1 1	1 1 0 0	1 1 0 0
9.	1 1 0 0	1 1 0 0	1 0 1 1	1 0 1 1
10.	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1
11.	1 1 1 0	0 1 0 1	1 1 1 0	0 1 0 1
12.	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

Table 3 shows that all input combinations are applied to the AND gate and full adder regarded as a basic cell except (a_i,b_i,y,z)=0001, 1001, 0011, and 0111. Similar to the discussion in [8], the patterns (0001) and (1001) can never appear at the inputs to a cell under normal operation. In order to apply all possible input combinations, the cell is modified such a way that the carry output of the input (0001) is changed from logical 0 to 1 [8]. As a result, the use of the combination (0001) can apply the combinations (0011) and (0111) to array. It should be noted, however, the use of the combination (1001) cannot apply these two combinations under the external constraints. Therefore, the following input patterns are added to Table 3 to apply all possible combinations.

13.	0 0 0 1	0 0 1 1	0 0 0 1	0 0 1 1
14.	0 0 1 1	0 0 0 1	0 0 1 1	0 0 0 1
15.	0 0 0 1	0 0 0 1	0 1 1 1	0 1 1 1
16.	0 1 1 1	0 1 1 1	0 0 0 1	0 0 0 1

According to the application of the labels of equation (7), the use of input combinations illustrated in Table 3 can generate the test patterns for a C-testable MCPM as shown in Table 4.

Table 4. Test Patterns for MCPM.

					TEST	
Test #	a	b	c	d	1	2
1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0	0
2	0 0 0 0	1 1 1 1	0 0 0 0	0 0 0 0	0	0
3	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0	0
4	1 0 1 0	1 1 1 1	1 0 1 0	1 1 1 1	0	0
5	0 0 0 0	0 0 0 0	1 1 1 1	0 0 0 0	1	1
6	0 0 0 0	1 1 1 1	1 1 1 1	0 0 0 0	1	1
7	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1	1
8	1 1 1 1	1 0 1 0	1 1 1 1	0 1 0 1	1	1
9	1 1 1 1	0 1 0 1	0 0 0 0	1 0 1 0	1	1
10	1 1 1 1	1 1 1 1	0 0 0 0	1 1 1 1	1	1
11	0 1 0 1	1 1 1 1	0 1 0 1	0 0 0 0	0	0
12	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	0	0
13	0 0 0 0	0 0 0 0	1 0 1 0	1 1 1 1	0	0
14	0 0 0 0	0 0 0 0	0 1 0 1	1 1 1 1	1	1
15	0 0 0 0	1 0 1 0	0 0 0 0	1 1 1 1	0	1
16	0 0 0 0	0 1 0 1	1 1 1 1	1 1 1 1	1	0

Applying these test patterns to array results that all 8 possible input combinations have been completely applied to the left-boundary cell, while the 15 possible input combinations listed in Table 3 have been completely applied to the rest of cells. In other words, the applied test patterns can detect any single faults that occur in a cell of the array. In addition, these test patterns can also be applied to detect any single stuck-at faults that occur in those AND gates and XOR gates connected to the left-boundary cells. Therefore, the following Theorem results.

Theorem 2.
The MCPM is C-testable with a test length of 16.

3.3. Design Evaluation.

Consider the extra hardware required for an n-by-n MCPM. Unlike the cells consisting of an AND gate and a full adder in the original CPM, the cells in the proposed CPM are modified as follows: Each left-boundary cell only consists of a full adder; the corresponding AND gate is seperated from the cell and connected to a XOR gate with a control signal; and other cells are redesigned such that it still retains the operation in the original design, but it changes the carry output of the input (1000) from 0 to 1; Therefore, the extra hardware for a n-by-n MCPM is (n-1)'s XOR gates and extra (2n+2)'s pins for control signals. In fact, the hardware required for the basic cell modification is negligible. Furthermore, since the extra XOR gates are not located at the critical path, thus the speed performance will not be degraded.

IV. Design of C-testable Carry-Save Array Multipliers

A 4-by-4 carry-save array multiplier (CSM) is illustrated in Figure 5(a). The array receives two 4-bit operands and produces an 8-bit product. The graph labeling technique is also applied to generate labels for the design of a C-testable CSM. Similar to Theorem 1 for MCPM, consider the following four vectors.

Vector 1: (V_1,V_2,V_3) Vector 2: (C_2,C_1,C_4)
Vector 3: (C_3,C_4,C_1) Vector 4: (V_4,V_3,V_2)

These four vectors are perfectly applied to array without requiring any extra hardware as shown in Figure 5(b). Consider a 4-by-4 modified carry-save array multiplier (MCSM), as shown in Figure 5(a). The c', c and d inputs of the cells in the leftmost diagonal are connected to logical 0 during multiplication. However, it is assumed that during testing these inputs are available as primary inputs to the array.

The input combinations for MCSM can be generated in a similar way as shown in Table 3 under the external constraints [10], and its test patterns are shown in Table 5.

Table 5. Test Patterns for MCSM.

Test #	a	b	c'	c	d
1.	0000	0000	0000	0000	0000
2.	0000	1111	0000	0000	0000
3.	1111	0000	0000	0000	0000
4.	0101	1111	1111	1010	1010
5.	1111	1010	0101	0000	1111
6.	1111	1111	0000	0000	1111
7.	0000	0000	1111	1111	0000
8.	0000	1111	1111	1111	0000
9.	1111	0000	1111	1111	0000
10.	1111	0101	1010	1111	0000
11.	1010	1111	0000	0101	0101
12.	1111	1111	1111	1111	1111
13.	0000	0000	1111	1010	1111
14.	0000	0000	0000	0101	1111
15.	1010	0000	1111	1010	1111
16.	0101	0000	0000	0101	1111

If we apply the test patterns of Table 5 to the MCSM, then all possible input combinations are applied to each cell without requiring any extra hardware. Therefore, the following Theorem results.

Theorem 3:
The modified carry-save array multiplier (CSM) is C-testable with a test length of 16.

V. Design of C-testable Baugh-Wooley Multipliers

The same testing methodology applied to MCSM is applied to Baugh-Wooley two's complement array multiplier (BWM) [3]. Consider the circuit schematic diagram, as shown in Figure 5, for modified Baugh-Wooley array multiplier (MBWM) [10]. The 16 test patterns shown in Table 6 can detect any signle faults in that array multiplier. This results

Theorem 4.
The modified Baugh-Wooley array multiplier is C-testable with a test length of 16.

Table 6. Test Patterns for MBWM.

Test #	a	b	d	s1	s2	s3	s4	e1
1.	00000	00000	0000	0	0	0	1	0
2.	00000	11111	0000	0	0	0	0	0
3.	01111	00000	0000	0	0	0	1	0
4.	10101	11111	1010	0	0	0	0	1
5.	11111	10101	1111	1	1	1	0	0
6.	01111	11110	1111	0	0	0	0	0
7.	10000	00000	0000	1	1	1	0	0
8.	10000	11111	0000	1	1	0	0	0
9.	11111	00000	0000	1	1	1	0	0
10.	11111	01010	0000	1	1	1	0	0
11.	01010	11110	0101	1	0	0	0	1
12.	11111	11111	1111	0	0	0	0	1
13.	10000	00000	1111	0	1	1	0	1
14.	00000	00000	1111	1	0	0	0	1
15.	11010	00000	1111	0	1	1	0	1
16.	00101	00000	1111	1	0	0	0	1

VI. Conclusions

The designs of C-testable carry-propagate array multipliers, carry-save array multipliers, and Baugh-Wooley array multipliers, are presented. Based on the graph labeling scheme [9], the test pattern generation procedure is discussed. The results of the study show that the above mentioned array multipliers require only 16 test patterns to detect any single faults. Although the extra control signals required during test mode may significantly increased the pin counts. a built-in self-test can alleviate the problem.

References

[1] Abraham, J.A., and V.K. Agarwal, "Test generation for digital systems," in **Fault-tolerant Computing, Theory and Techniques**, edited by D.K. Pradhan, Prentice-Hall, Englewood Cliffs, N.J., 1986.

[2] Lala, P.K., **Fault Tolerant & Fault Testable Hardware Design**, Prentice-Hall, 1985.

[3] Hwang, K., Computer Arithmetic, Principles, Architecture, and Design, Wiley, New York, 1979.

[4] Waser, S., "High speed monolithic multipliers for real-time digital signal processing," Computer, pp.19-28, Oct. 1978.

[5] Kautz, W.H., "Testing for faults in cellular logic arrays," Proc. 8th Symp. Switching Automata Theory, 1967, pp.161-174.

[6] Friedman, A.D., "Easily testable iterative systems," IEEE Trans. on Computers, Vol.C-22, pp.1061-1064, Dec. 1973.

[7] Parthasarathy, R., and S.M. Reddy, "A testable design of iterative logic arrays," IEEE Trans. on Circuits and Systems, Vol.CAS-28, pp.2037-1045. Nov. 1981.

[8] Shen, J.P., and F.J. Ferguson, "The design of easily testable array multipliers," IEEE Trans. on Computers, Vol.C-33, No.6, pp.554-560, June 1984.

[9] Chatterjee, A., and J.A. Abraham, "Test generation for arithmetic unit by graph labeling," Proc. of 17th Fault-Tolerant Computing Symp., Pittsburgh, PA, pp.284-289, July 1987.

[10] Wey, C.L., and S.M. Chang, "Test Generaton for C-testable Array Multipliers," Technical Report, Division of Engineering Research, Michigan State University, East Lansing, MI.

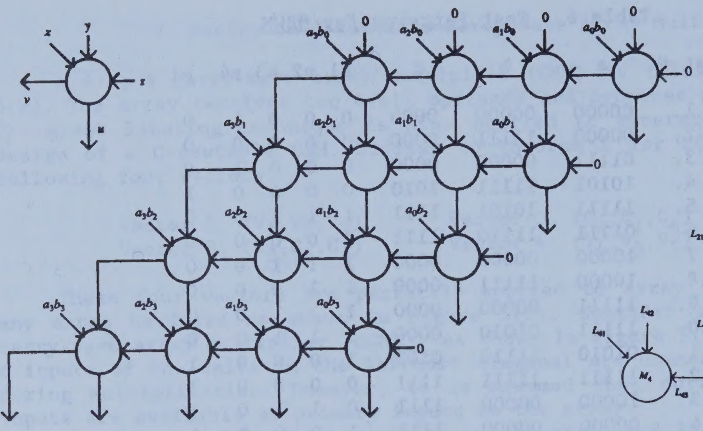

x
y
z
v
u
a_3b_0
a_2b_0
a_1b_0
a_0b_0
0
a_3b_1
a_2b_1
a_1b_1
a_0b_1
a_3b_2
a_2b_2
a_1b_2
a_0b_2
a_3b_3
a_2b_3
a_1b_3
a_0b_3

Figure 1. A 4-by-4 Carry-Propagate Array Multiplier [9].

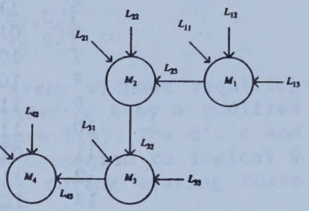
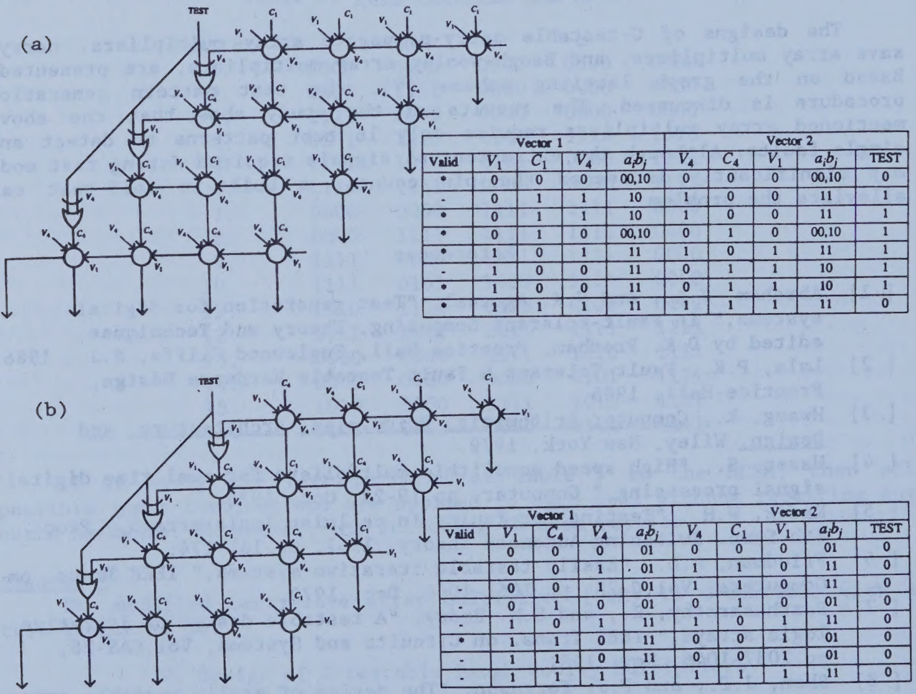

L_{22}
L_{21}
L_{23}
M_2
L_{11}
L_{12}
M_1
L_{13}
L_{42}
L_{41}
M_4
L_{43}
L_{31}
L_{32}
M_3
L_{33}

Figure 3. Four Baisc Labels.


(a)
TEST
V_1
C_1
V_4
C_4
(b)

Vector 1					Vector 2				
Valid	V_1	C_1	V_4	a_ib_j	V_4	C_4	V_1	a_ib_j	TEST
•	0	0	0	00,10	0	0	0	00,10	0
-	0	1	1	10	1	0	0	11	1
•	0	1	1	10	1	0	0	11	1
•	0	1	0	00,10	0	1	0	00,10	1
•	1	0	1	11	1	0	1	11	1
-	1	0	0	11	0	1	1	10	1
•	1	0	0	11	0	1	1	10	1
•	1	1	1	11	1	1	1	11	0

Vector 1					Vector 2				
Valid	V_1	C_4	V_4	a_ib_j	V_4	C_1	V_1	a_ib_j	TEST
-	0	0	0	01	0	0	0	01	0
•	0	0	1	01	1	1	0	11	0
-	0	0	1	01	1	1	0	11	0
-	0	1	0	01	0	1	0	01	0
-	1	0	1	11	1	0	1	11	0
•	1	1	0	11	0	0	1	01	0
-	1	1	0	11	0	0	1	01	0
-	1	1	1	11	1	1	1	11	0

Figure 2. (a) Label L, (b) Lable L' for Carry-Propagate Array Multiplier [9].

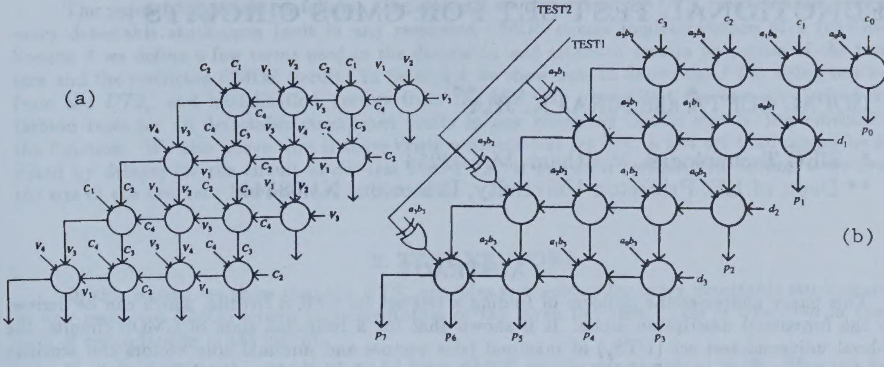

(a)
TEST2
TEST1
(b)
p_0
p_1
p_2
p_3
p_4
p_5
p_6
p_7
d_0
d_1
d_2
d_3
c_0
c_1
c_2
c_3

Figure 4. A 4-by-4 CPM: (a) Labeling, and (b) Schematic Diagram.

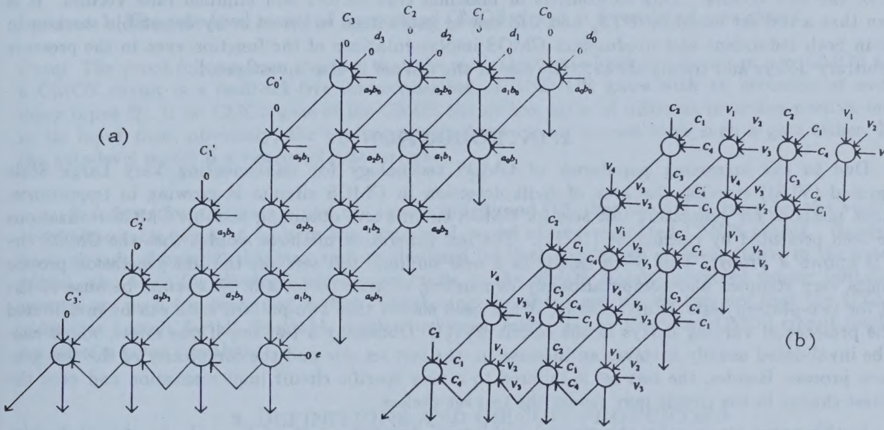

(a)
C_0
C_1
C_2
C_3
d_0
d_1
d_2
d_3
C_0'
C_1'
C_2'
C_3'
e
(b)

Figure 5. A 4-by-4 CSM: (a) Schematic Diagram, and (b) Labeling.

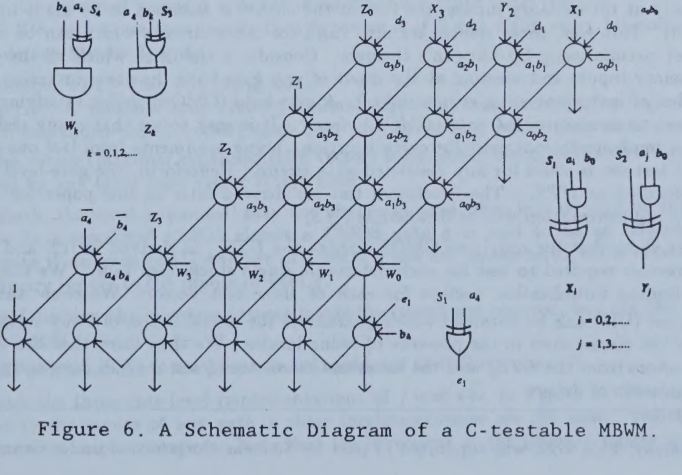

W_k
Z_k
k = 0,1,2,....
Z_0
X_3
Y_2
X_1
Z_1
Z_2
Z_3
W_3
W_2
W_1
W_0
e_1
b_4
S_1
a_4
X_i
Y_j
i = 0,2,....
j = 1,3,....

Figure 6. A Schematic Diagram of a C-testable MBWM.

A FUNCTIONAL TEST SET FOR CMOS CIRCUITS

GOPAL GUPTA* and NIRAJ K. JHA**

* SILC Technologies, Waltham, MA 02254
** Dept. of EE, Princeton University, Princeton, NJ 08544

ABSTRACT

This paper addresses the problem of finding a test set for CMOS circuits, which can be derived from the functional description alone. It is shown that for a restricted class of CMOS circuits, the gate-level universal test set (UTS_g) of maximal false vectors and minimal true vectors can sensitize every detectable stuck-open fault in the circuit. A universal initialization set (UIS) is defined which can also be derived from just the functional description and which contains initialization vectors for each of the test vectors. This set consists of maximal true vectors and minimal false vectors. It is shown that a test set based on UTS_g and UIS can be guaranteed to detect every detectable stuck-open fault in both redundant and irredundant CMOS implementations of the function even in the presence of arbitrary delays and timing-skews. The size of the test set is also investigated.

1. INTRODUCTION

Due to the increasing popularity of CMOS technology for implementing Very Large Scale Integrated (VLSI) circuits, the area of fault detection in CMOS circuits is growing in importance. Various methods for generating test sets for CMOS circuits and obtaining testable CMOS realizations have been presented by researchers [1]-[11]. The test generation methods assume that the CMOS circuit is known *a priori*. This often results in a near minimal test set, but the test generation process becomes very complex and computationally demanding -- more so for CMOS circuits because of the need for two-pattern tests. Furthermore, it has been shown that two-pattern tests can be invalidated in the presence of varying delays in the circuit [8]-[11]. Obtaining a test set, if one exists, which cannot be invalidated usually involves an increase in the test set size and the complexity of the test generation process. Besides, the test set is tailored to a very specific circuit implementation and even the slightest change in the circuit may render the test set useless.

In this paper we consider the general problem of deriving a test set from the functional description only. Our fault model includes all detectable single stuck-open faults. Akers [12] and Reddy [13] have already given solutions to this problem with respect to gate-level circuits. They showed that the universal test set, comprising of the maximal false vectors and the minimal true vectors, tests for all detectable stuck-at faults in a particular class of gate-level circuits. In such a circuit the inverters can be present only at the primary inputs, the rest of the circuit is assumed to be made up of AND and OR gates only. However, these results are also valid for other circuits which can be reduced to the above class of circuits using De Morgan's theorem. Consider a circuit in which all the paths starting from the primary inputs and meeting at the input of any gate have the same inversion parity (either an odd number of inversions or an even number). A gate-level (CMOS) circuit satisfying this property will be referred to as a *restricted gate* (*CMOS*) *circuit*. It is easy to see that many restricted gate or CMOS circuit implementations exist for every function. Using arguments from [13] one can show that the universal test set is valid for any restricted gate circuit. Henceforth, the gate-level universal test set will be denoted as UTS_g. The functional test set derived later in this paper for any restricted CMOS circuit implementation will be denoted as FTS_c.

We first show that for restricted CMOS circuits the UTS_g, as derived in [12] and [13], provides all the test vectors required to test for each detectable single stuck-open fault. We then consider the problem of finding initialization vectors for each of these test vectors. We show that a universal initialization set (UIS) can be obtained which guarantees the initialization of every initializable stuck-open fault in the circuit even in the presence of redundancies. We then show that the FTS_c comprising of test vectors from the UTS_g and the initialization vectors from the UIS does not get invalidated even in the presence of delays.

Acknowledgement: This work was supported in part by Siemens Corporation under Grant 170-4051.

The paper is organized as follows. In Section 2 we show that the UTS_g contains test vectors for every detectable stuck-open fault in any restricted CMOS circuit implementation of a function. In Section 3 we define a few terms used in the discussion and establish certain properties of the test vectors and the restricted CMOS circuit. In Section 4 we show how to derive the FTS_c using test vectors from the UTS_g and initialization vectors from the UIS. We prove that the test set derived in this fashion tests for all detectable stuck-open faults in any restricted CMOS circuit implementation of the function. We also prove that if there exists a robust test set (i.e., a test set that cannot be invalidated by delays) for the circuit under test then FTS_c is one such test set. In Section 5 we evaluate the size of the test set. We conclude in Section 6.

2. TEST VECTORS

In this section we show that the UTS_g provides test vectors for every detectable stuck-open fault in any restricted CMOS circuit implementation of the given function. This is true even in the presence of redundancies in the circuit.

A method to get the gate-level model for a CMOS circuit was presented in [2]. This can be used to arrive at the following lemma.

Lemma 1: The gate-level model of a restricted CMOS circuit is a restricted gate circuit.

Proof: The proof follows from the fact that the equivalent gate-level sub-circuit of any CMOS gate in a CMOS circuit is a feedback-free interconnection of AND/OR gates with an inversion at each primary input [2]. If no CMOS gate of the CMOS circuit has paths of different inversion parities meeting at its input then, obviously, the equivalent gate-level circuit cannot have such a gate either. Hence, the gate-level model is a restricted gate circuit.

□

It follows from Lemma 1 and the results in [12] and [13] that the UTS_g provides test vectors for every detectable stuck-at fault in the gate-level model of any restricted CMOS circuit. Reddy et al have already shown [2] that any test set designed for detecting single stuck-at faults in the gate-level model, also tests for all single stuck-open faults in the CMOS circuit, provided proper initialization vectors are used for every test vector. Combining the above results, we can say that the UTS_g provides test vectors for all detectable single stuck-open faults in any restricted CMOS circuit implementation of a given function.

3. DEFINITIONS AND CIRCUIT PROPERTIES

In this section we define different terms used in the discussion and establish certain properties of restricted CMOS circuits.

Definition 1: A true (false) test vector is one for which the circuit output in the fault-free case is logic 1(0).

Definition 2: An expanded vector is a vector in an expanded truth table [12],[13].

Definition 3: If δ_1 and δ_2 are two vectors then δ_2 covers δ_1 if δ_2 has 1's in every bit-position in which δ_1 has 1's. We denote this by $\delta_2 \geq \delta_1$.

For example, $110 \geq 100$, but $110 \not\geq 001$.

Definition 4 [13]: A minimal true vector (minimal expanded true vector) of a logic function is the input vector (expanded input vector) that does not cover any other true vector except itself. A maximal false vector (maximal expanded false vector) is the input vector (expanded input vector) that is not covered by any other false vector except itself.

Henceforth, the word 'expanded' may not always be used when it is implied by the context.

Definition 5: In a restricted CMOS circuit a CMOS gate g is said to be an odd-numbered (even-numbered) gate if an even (odd) number of CMOS gates are encountered while traversing any path from the primary inputs to the inputs of gate g.

In this definition the inverters present inside the circuit are counted, but not the ones present at the primary inputs.

We will now establish a few properties of restricted CMOS circuits.

Consider the three gate-level implementations of $f = ab + \bar{b}c$ in Figure 1. The inversion parities of the paths to the inputs of any gate in these implementations are the same. Hence, these are restricted gate circuit implementations. Note that the inversions at the primary inputs are ignored while determining the inversion parity. The CMOS implementations of the above function which correspond to these three gate-level circuits are, obviously, restricted CMOS circuits.

Since every CMOS gate introduces an inversion, it means that all paths incident at any CMOS gate in a restricted CMOS circuit have either an even number or an odd number of CMOS gates. In the discussion that follows, we shall assume that the number of CMOS gates in each path from the primary inputs to the circuit output is even. This does not result in any loss of generality, since by applying De Morgan's theorem we can always transfer an inversion at the circuit output to the primary inputs. The test set of the original circuit remains valid even for the modified circuit obtained after the inversion transfer. But the number of CMOS gates in each path to the circuit output can be changed from odd to even by this method. The following remarks follow from these observations:

Remark 1 : A true (false) test vector sensitizes stuck-open faults in the nMOS (pMOS) networks of odd-numbered gates and the pMOS (nMOS) networks of even-numbered gates.

This can be seen by retracing the path from the circuit output to the primary inputs, knowing that a true (false) test vector activates the pMOS (nMOS) network of the output CMOS gate since the output is logic 1 (0).

Remark 2 : If δ_1 and δ_2 are two expanded vectors of a function f , and $\delta_1 \geq \delta_2$, then in the transition $\delta_2 \longrightarrow \delta_1$ the output of an even-numbered gate in any restricted CMOS circuit implementation can only change from 0 to 1, while the output of an odd-numbered gate can only change from 1 to 0.

We know that in a restricted CMOS circuit the output of an even-numbered (odd-numbered) gate can only be reached through an even (odd) number of inversions on any path from the primary inputs. Hence, it follows that since input literals either stay at 0 or change from 0 to 1 in the transition $\delta_2 \longrightarrow \delta_1$, the outputs of the even-numbered gates would either stay at 0 or go from 0 to 1, and the outputs of the odd-numbered gates would either stay at 1 or go from 1 to 0. Analogous results can be obtained for the transition $\delta_1 \longrightarrow \delta_2$. In this case, the output of each gate can only vary in the complementary manner.

Since the circuit output is taken from an even-numbered CMOS gate the circuit is monotonic in the expanded vectors. In other words, if δ_1 and δ_2 are two expanded vectors of a function f and $\delta_1 \geq \delta_2$ then $f(\delta_1) \geq f(\delta_2)$, else if $\delta_1 \leq \delta_2$ then $f(\delta_1) \leq f(\delta_2)$.

4. DERIVING FUNCTIONAL TEST SETS

In an irredundant restricted CMOS circuit, for every single stuck-open fault sensitized by a vector $\delta_1 \in UTS_g$ there exists another vector $\delta_2 \in UTS_g$ which sensitizes the stuck-open fault in the corresponding transistor (the transistor fed by the same input) in the complementary network of that gate. The UTS_g, therefore, provides all the test and initialization vectors required for testing a restricted CMOS circuit for single stuck-open faults. It is sufficient to obtain two-pattern tests by preceding every minimal true vector with every maximal false vector and, vice-versa, in order to derive a test set which detects all single stuck-open faults in any irredundant restricted CMOS circuit implementation of the given function. The resultant test set, however, can be invalidated in the presence of arbitrary delays. Besides, the test set is valid only for irredundant implementations.

In this section we look at the problem of deriving a test set which can detect all detectable single stuck-open faults in any restricted CMOS circuit implementation (redundant or irredundant) of a given function. Furthermore, we show that this test set does not get invalidated in the presence of delays. In a redundant circuit, not every stuck-open fault can be sensitized to the circuit output. We do have the guarantee that the vectors in the UTS_g will sensitize every sensitizable fault to the circuit output. But there is no guarantee that for every CMOS gate in a redundant circuit a single stuck-open fault in a transistor in the nMOS network and in the corresponding transistor in the pMOS network can both be sensitized to the circuit output.

For example, consider the function $f = ab + ac$. A redundant CMOS realization of this function is given in Figure 2. The UTS_g for this function is given in Table 1. Let a transistor in the pMOS network corresponding to a transistor γ in the nMOS network be referred to as γ^c . Now a stuck-open fault in transistor 5^c is sensitized by β_2, but neither α_1 nor α_2 establishes a conduction path through transistor 5. If we confine ourselves to the UTS_g for initialization vectors, this fault will be left undetected.

Table 1: UTS_g for $f = ab + ac$

a b c	f	label
1 1 0	1	α_1
1 0 1	1	α_2
1 0 0	0	β_1
0 1 1	0	β_2

We will give a method for deriving a set of vectors, not necessarily in the UTS_g, which will guarantee the initialization of the circuit for every detectable stuck-open fault. Before doing so we define a few terms.

Definition 6: An expanded true vector κ_i is a maximal expanded true vector if there is no other expanded true vector κ_j such that $\kappa_j \geq \kappa_i$. Analogously, an expanded false vector ξ_i is a minimal expanded false vector if there is no other expanded false vector ξ_j such that $\xi_i \geq \xi_j$.

Definition 7: The universal initialization set (UIS) is the set of maximal expanded true vectors and minimal expanded false vectors. We will denote the sub-set of true vectors by I_1 and the sub-set of false vectors by I_0.

For example, for the function implemented in the circuit in Figure 2 the UIS is given in Table 2.

Table 2: UIS for $f = ab + ac$

a b c	f	label
1 1 1	1	κ_1
0 0 0	0	ξ_1

The initialization of the circuit by the UIS is considered in the following theorem.

Theorem 1: The set of maximal expanded true vectors I_1 contains initialization vectors, if they exist, for every false test vector in UTS_g, while the set of minimal expanded false vectors I_0 contains initialization vectors, if they exist, for every true test vector in UTS_g for any restricted CMOS circuit implementation of the given function.

Proof: The proof is given for a true test vector only. The proof for a false test vector follows in an analogous fashion. Suppose I_0 does not have initialization vectors for every true test vector, i.e., there exists a vector σ_i which can serve as an initialization vector for some stuck-open fault sensitized by a true test vector, but no vector belonging to I_0 can do so. We know from Remark 1 that a true test vector tests for stuck-open faults in pMOS networks of even-numbered CMOS gates and nMOS networks of odd-numbered CMOS gates in any sensitized path. To initialize the output of a CMOS gate for a fault in one of its networks we need to establish a conducting path through the complementary network, i.e., the nMOS networks of even-numbered gates and the pMOS networks of odd-numbered gates in the sensitized path. Since σ_i is not in I_0 there must be a vector $\xi_i \epsilon I_0$ such that $\xi_i \leq \sigma_i$. Thus, in the transition $\sigma_i \rightarrow \xi_i$ the only changes in the different bits can be from 1 to 0. From Remark 2, the only resultant change can be from 1 to 0 at the outputs of even-numbered gates and from 0 to 1 at the outputs of odd-numbered gates. Thus, the output of the faulty CMOS gate initialized by σ_i would also be initialized by ξ_i. This contradicts our initial assumption and, hence, the theorem follows.

□

For the circuit in Figure 2, if we use the UIS given in Table 2, the stuck-open fault in 5^c can now be sensitized by β_2 from Table 1 and initialized by κ_1 from Table 2.

The functional test set FTS_c is derived by obtaining two-pattern tests by preceding every true (false) test vector from the UTS_g by every false (true) initialization vector from the UIS. For the function $f = ab + ac$ considered earlier, the FTS_c is given below.

Table 3: FTS_c for $f = ab + ac$

a b c	f
1 1 1	1
1 0 0	0
1 1 1	1
0 1 1	0
0 0 0	0
1 1 0	1
0 0 0	0
1 0 1	1

Definition 8: Let δ_1 and δ_2 be two expanded vectors in the expanded truth table of a function f. Then $(\delta_1 + \delta_2)$ is the vector obtained by a bitwise OR of the vectors δ_1 and δ_2.

For example, (011 + 101) is the vector 111.

Definition 9: Let δ_1 and δ_2 be two expanded vectors in the expanded truth table of a function f. Then $(\delta_1 . \delta_2)$ is the vector obtained by a bitwise AND of the vectors δ_1 and δ_2.

For example, (011 . 101) is the vector 001.

Lemmas 2 and 3 serve to establish a basis for ensuring the robustness of a two-pattern test by comparing a particular two-pattern test with another two-pattern test which is known to be robust.

Lemma 2: For a restricted CMOS circuit let δ_2 be an expanded true test vector and δ_1 the corresponding initialization vector such that (δ_1, δ_2) is a robust two-pattern test for detecting some stuck-open fault ϕ_1. If (δ_1', δ_2') is another two-pattern test for detecting ϕ_1 such that $(\delta_1' + \delta_2') \leq (\delta_1 + \delta_2)$, then (δ_1', δ_2') is also robust.

Proof: From Remark 1, to invalidate a two-pattern test comprising of a true test vector, a spurious conduction path must get activated through the nMOS network of an odd-numbered gate or the pMOS network of an even-numbered gate on the sensitized path. This can happen if there exists a vector δ_3, $(\delta_1 . \delta_2) \leq \delta_3 \leq (\delta_1 + \delta_2)$, that establishes such a spurious conduction path. Since (δ_1, δ_2) is a robust two-pattern test, clearly, there does not exist such a vector δ_3. In particular, the vector $(\delta_1 + \delta_2)$ does not establish such a spurious conduction path. Since $(\delta_1' + \delta_2') \leq (\delta_1 + \delta_2)$, we know from Remark 2 that the only additional conduction paths that $(\delta_1' + \delta_2')$ can establish apart from the ones established by $(\delta_1 + \delta_2)$ are through the nMOS networks of even-numbered gates and the pMOS networks of odd-numbered gates. Thus, $(\delta_1' + \delta_2')$ cannot establish a spurious conduction path. Hence, again from Remark 2, there also does not exist any vector $\delta_3' \leq (\delta_1' + \delta_2')$ that establishes a spurious conduction path. (δ_1', δ_2'), therefore, is a robust two-pattern test for detecting ϕ_1.

□

Lemma 3: For a restricted CMOS circuit let δ_2 be an expanded false test vector and δ_1 the corresponding initialization vector such that (δ_1, δ_2) is a robust two-pattern test for detecting a stuck-open fault ϕ_1. If (δ_1', δ_2') is another two-pattern test for detecting ϕ_1 such that $(\delta_1' . \delta_2') \geq (\delta_1 . \delta_2)$, then (δ_1', δ_2') is also robust.

Proof: Analogous to the proof of Lemma 2.

□

Lemma 4: If R is a restricted CMOS circuit which is monotonic in the expanded vectors in the fault-free case, then R remains monotonic even in the presence of a stuck-open fault.

Proof: Let g be the CMOS gate with the stuck-open fault. We recall that the monotonic property for the restricted CMOS circuits considered in Section 3 follows directly from the fact that, if δ_1 and δ_2 are two expanded vectors such that $\delta_1 \geq \delta_2$ then in the transition $\delta_1 \rightarrow \delta_2$ the output of an even-numbered gate can only change from 1 to 0, while the output of an odd-numbered gate can only change from 0 to 1. Analogously, in the transition $\delta_2 \rightarrow \delta_1$ the output of an even-numbered gate can only change from 0 to 1, while the output of an odd-numbered gate can only change from 1 to 0. We need to only show that the above property holds even when the stuck-open fault is present.

Let the faulty gate g be an even-numbered gate. Consider the transition $\delta_1 \rightarrow \delta_2$. All inputs to the gate g are the outputs of odd-numbered gates. Besides, all gates in the paths from the primary inputs to the inputs of gate g are fault-free and, hence, from Remark 2, the only transition at the inputs of the gate g can be 0→1. Clearly, even in the presence of a stuck-open fault the only additional conduction paths established in g during the transition $\delta_1 \rightarrow \delta_2$ are through the nMOS network.

Hence, the only transition at the output of g can be $1 \rightarrow 0$. Now consider the transition $\delta_2 \rightarrow \delta_1$. The inputs to gate g can only change from 1 to 0 since all these inputs are the outputs of odd-numbered gates. The only additional conduction paths established as a result of this transition are through the pMOS network of g. Hence, the only transition at the output of g can be $0 \rightarrow 1$. The above arguments prove that a faulty even-numbered gate preserves the property that ensures monotonicity. The proof for the case where the faulty gate is odd-numbered follows in an analogous fashion.

□

We can now prove that FTS_c is a robust test set.

Theorem 2: If there exists a robust test set for a restricted CMOS circuit implementation of the given function, then FTS_c, comprising of test vectors from the UTS_g and the initialization vectors from the UIS, is one such robust test set.

Proof: Let V be a set of robust two-pattern tests derived for all single stuck-open faults in the restricted CMOS circuit under test. We shall prove that for every robust two-pattern test in V, there exists a robust two-pattern test in FTS_c which tests for the same stuck-open faults.

Let δ_2 be an expanded true test vector and (δ_1,δ_2) a robust two-pattern test in V for a set of single stuck-open faults. From Remark 1, δ_2 tests for stuck-open faults in the nMOS networks of odd-numbered gates and the pMOS networks of even-numbered gates on the sensitized path. Since δ_1 is the initialization vector for δ_2, it must establish conduction paths through pMOS networks of odd-numbered gates and nMOS networks of the even-numbered gates on the sensitized path. From the definitions of the UTS_g and the UIS, there must exist an expanded true vector $\alpha \in UTS_g$ and an expanded false vector $\xi \in UIS$, such that $\alpha \leq \delta_2$ and $\xi \leq \delta_1$. It can be easily shown that α sensitizes all stuck-open faults sensitized by δ_2. In the transition $\alpha \rightarrow \delta_2$ the output of an odd-numbered gate can only change from 1 to 0 and that of an even-numbered gate from 0 to 1. Thus, if α does not test for some stuck-open fault, i.e., the circuit output is 1 even in the presence of the fault, then, from Lemma 4, δ_2 also cannot test for that fault. Further, since $\xi \leq \delta_1$, in the transition $\delta_1 \rightarrow \xi$ the output of an odd-numbered gate can only change from 0 to 1 and that of an even-numbered gate from 1 to 0. As a result, ξ initializes the circuit for each of the faults that δ_1 initializes the circuit for. Since $\alpha \leq \delta_2$ and $\xi \leq \delta_1$, $(\alpha+\xi) \leq (\delta_1+\delta_2)$. From Lemma 2, (ξ,α) is also a robust two-pattern test for all the stuck-open faults detected by (δ_1,δ_2).

Now assume that (δ_1,δ_2) is a robust two-pattern test in V in which δ_2 is an expanded false test vector. There must exist an expanded false vector $\beta \in UTS_g$ and an expanded true vector $\kappa \in UIS$, such that $\beta \geq \delta_2$ and $\kappa \geq \delta_1$. Then, using Lemma 3, we can prove in an analogous fashion that (κ,β) is a robust two-pattern test for all the stuck-open faults detected by (δ_1,δ_2).

□

5. SIZE OF THE TEST SET

In this section we compare the size of the functional test set FTS_c, comprising of test vectors from the UTS_g and initialization vectors from the UIS, with the minimal size test set for some given implementation of a function. In the discussion ahead we denote the set of minimal (maximal) true vectors by T_1 (I_1) and the set of maximal (minimal) false vectors by T_0 (I_0). In accordance with the definitions of UTS_g and UIS, $UTS_g = T_1 \bigcup T_0$ and $UIS = I_1 \bigcup I_0$.

Let the sets I_1, T_1, I_0 and T_0 be given as follows:

$$\begin{aligned} I_1 &= \{\alpha_1,\alpha_2,..,\alpha_i,\kappa_1,\kappa_2,..,\kappa_j\} \\ T_1 &= \{\alpha_1,\alpha_2,..,\alpha_i,\delta_1,\delta_2,..,\delta_k\} \\ I_0 &= \{\beta_1,\beta_2,..,\beta_l,\xi_1,\xi_2,..,\xi_m\} \\ T_0 &= \{\beta_1,\beta_2,..,\beta_l,\gamma_1,\gamma_2,..,\gamma_n\} \end{aligned}$$

such that $S_0 = I_0 \bigcap T_0 = \{\beta_1,\beta_2,..,\beta_l\}$, $S_1 = I_1 \bigcap T_1 = \{\alpha_1,\alpha_2,..,\alpha_i\}$, $S_2 = \{\kappa_1,\kappa_2,..,\kappa_j\}$, $S_3 = \{\delta_1,\delta_2,..,\delta_k\}$, $S_4 = \{\xi_1,\xi_2,..,\xi_m\}$, and $S_5 = \{\gamma_1,\gamma_2,..,\gamma_n\}$. Hence, $I_1 = S_1 \bigcup S_2$, $T_1 = S_1 \bigcup S_3$, $I_0 = S_0 \bigcup S_4$, and $T_0 = S_0 \bigcup S_5$. Theorem 3 gives the lowerbound on the size of the FTS_c.

Theorem 3: The minimal attainable size of the FTS_c, denoted as $|FTS_c|_{\min}$, is equal to

$$2(|I_0|.|T_1|+|I_1|.|T_0|)-|S_1|.\min\{|S_4|,|S_5|\}-|S_0|.\min\{|S_2|,|S_3|\}-2|S_0|.|S_1|+p,$$

where p = 1, if S_0 and S_1 are non-empty, and 0 otherwise.

Proof: There are four cases to be considered.

Case 1: $S_0 = S_1 =$ null.

Hence, we have to prove that

$$|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0|)$$

For this case there is no vector which can be used both as a test vector of some two-pattern test

and an initialization vector of the next two-pattern test. Since there are $(|I_0|.|T_1| + |I_1|.|T_0|)$ initialization-test pairs, the proof follows directly.
Case 2: $S_0 =$ null, $S_1 \neq$ null.

Therefore, we have to prove that

$$|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0|) - |S_1|.\min\{|S_4|, |S_5|\}$$

First assume that $|S_4| \leq |S_5|$. Consider the following sequence of vectors:

$$\xi_1\alpha_1\gamma_1, \xi_1\alpha_2\gamma_1, .., \xi_1\alpha_i\gamma_1, \xi_2\alpha_1\gamma_2, \xi_2\alpha_2\gamma_2, .., \xi_2\alpha_i\gamma_2, .., \xi_m\alpha_1\gamma_m, \xi_m\alpha_2\gamma_m, .., \xi_m\alpha_i\gamma_m$$

This sequence uses a vector $\alpha_q \epsilon S_1$ as both a test vector and an initialization vector as many times as possible. There are $|S_1|.|S_4|$ cases in which this can be done. Also, there are no other cases in which this is possible. Hence, from the above arguments, $|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0|) - |S_1|.|S_4|$. Similarly, if we assumed that $|S_4| > |S_5|$, then it can be shown that $|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0|) - |S_1|.|S_5|$. Combining these two results, $|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0| - |S_1|.\min\{|S_4|, |S_5|\}$.
Case 3: $S_0 \neq$ null, $S_1 =$ null.

For this case we have to prove that

$$|FTS_c|_{\min} = 2(|I_0|.|T_1| + |I_1|.|T_0|) - |S_0|.\min\{|S_2|, |S_3|\}$$

The proof is exactly similar to the proof of Case 2.
Case 4: $S_0 \neq$ null, $S_1 \neq$ null.

For this case we have to prove that $|FTS_c|_{\min}$ is equal to

$$2(|I_0|.|T_1| + |I_1|.|T_0|) - |S_1|.\min\{|S_4|, |S_5|\} - |S_0|.\min\{|S_2|, |S_3|\} - 2|S_0|.|S_1| + 1$$

Hence, we have to show that apart from the reduction in $|FTS_c|$ shown possible in the proofs of Case 2 and Case 3, a further reduction of $2|S_0|.|S_1| - 1$ is possible.

By preceding every $\alpha_r \epsilon S_1$ ($\beta_s \epsilon S_0$) with every $\beta_s \epsilon S_0$ ($\alpha_r \epsilon S_1$) we can obtain the following two sequences of two-pattern tests:

(1) $\beta_1\alpha_1, .., \beta_l\alpha_1, \beta_1\alpha_2, .., \beta_l\alpha_2, .., \beta_1\alpha_i, .., \beta_l\alpha_i$
(2) $\alpha_1\beta_1, .., \alpha_i\beta_1, \alpha_1\beta_2, .., \alpha_i\beta_2, .., \alpha_1\beta_l, .., \alpha_i\beta_l$

The second sequence may be rearranged as

$$\alpha_1\beta_2, .., \alpha_1\beta_l, \alpha_1\beta_1, \alpha_2\beta_2, .., \alpha_2\beta_l, \alpha_2\beta_1, .., \alpha_i\beta_2, .., \alpha_i\beta_l, \alpha_i\beta_1$$

This rearranged sequence can be merged with the first sequence to arrive at the following sequence of vectors:

$$\beta_1\alpha_1..\beta_l\alpha_1\beta_1\alpha_2..\beta_l\alpha_2..\beta_1\alpha_i..\beta_l\alpha_i\beta_1$$

This sequence of vectors can be looked on as either Sequence 1 with β_1 added at the end or as the rearranged Sequence 2 with β_1 added at the beginning. Hence, it contains all the two-pattern tests contained in Sequence 1 or Sequence 2. Since the length of each of Sequence 1 and 2 is $2|S_0||S_1|$, it follows that the length of the total test set can be further reduced by $(2|S_0||S_1| - 1)$. We cannot do any better since every vector in S_0 or S_1 that could be used as both a test vector and an initialization vector has been used as such.

□

The question naturally arises as to how the minimal $|FTS_c|$ compares with the minimal size test set for any particular CMOS implementation of the given function. In other words, what is the price we pay in terms of the increased test set size for obtaining a functional test set. We have found that for a unate function $|FTS_c|$ is closer to the size of the minimal robust test set required for any particular restricted CMOS circuit implementation of the function.

Consider, for example, the implementation of the function $f = ab + bc$ as shown in Figure 2. The minimal size of the test set for this circuit is 6 while the minimal size of the robust test set for this circuit is 7. $|FTS_c|$ for this function is 2(2*1 + 2*1) = 8. Even for the simple irredundant NAND-NAND CMOS implementation of this function the minimal robust test set has 7 vectors in it.

As the binateness of the circuit increases, however, $|FTS_c|$ increases rather sharply. But the corresponding minimal size test set for any particular implementation of the function also registers a significant increase, more so when the implementation is redundant and we wish to ensure robustness.

Consider a totally binate function $f = a.\overline{b} + b.c + \overline{a}.\overline{c}$. Using the results of Theorem 3, it can be shown that $|FTS_c|$ = 2(6*2 + 2*6) - 2*2*6 + 1 = 25. However, the length of the minimal size robust test set for an irredundant NAND-NAND CMOS implementation of this function is 12.

Thus, we see that the method presented in this paper results in functional test sets which are near minimal for circuits which implement functions having a low degree of binateness. As the binateness of the function increases, the price paid in terms of a larger test set increases. However, as can be seen from the method presented in [8], the test generation process for generating a robust test set for any CMOS circuit is extremely time-consuming. Here the inherent robustness and ease of test generation of the functional test set give it a substantial edge. Of course, another advantage of having a functional test set is that a change in the implementation does not render the test set useless.

6. CONCLUSION

In this paper we presented a method for generating functional test sets for CMOS circuits. It was shown that the UTS_g sensitizes every sensitizable fault in any restricted CMOS circuit implementation of the function. The UIS, comprising of maximal true vectors and minimal false vectors, was defined, and it was shown that it provides all the initialization vectors needed to initialize the circuit even in the presence of redundancies. Further, the functional test set FTS_c, comprising of the test vectors from the UTS_g and the initialization vectors from the UIS, does not get invalidated even in the presence of arbitrary delays. The method given here presents a choice between a smaller, but computationally-demanding circuit-dependent test set and a larger, but easily-obtained and robust functional test set.

REFERENCES

[1] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell Syst. Tech. J.*, vol. 57, no. 2, pp. 1449-1474, May-June 1978.

[2] S. M. Reddy, V. D. Agrawal and S. K. Jain, "A gate-level model for CMOS combinational logic circuits with application to fault detection," in Proc. *Design Automation Conf.*, pp. 504-509, June 1984.

[3] R. Chandramouli, "On testing stuck-open faults," in Proc. *Int. Symp. Fault-Tolerant Comput.*, Milan, Italy, pp. 258-265, June 1983.

[4] K. W. Chiang and Z. G. Vranesic, "On fault detection in CMOS logic networks," in Proc. *Design Automation Conf.*, Miami Beach, FL, pp. 50-56, June 1983.

[5] Y. M. El-Ziq, "Automatic test generation for stuck-open faults in CMOS VLSI," in Proc. *Design Automation Conf.*, Nashville, TN, pp. 347-354, June 1981.

[6] Y. M. El-Ziq and R. J. Cloutier, "Functional-level test generation for stuck-open faults in CMOS VLSI," in Proc. *Int. Test Conf.*, Philadelphia, PA, pp. 536-546, Oct. 1981.

[7] S. K. Jain and V. D. Agrawal, "Test generation for MOS circuits using D-algorithm," in Proc. *Design Automation Conf.*, Miami Beach, FL, pp. 64-70, June 1983.

[8] S. M. Reddy, M. K. Reddy and V. D. Agrawal, "Robust tests for stuck-open faults in CMOS combinational logic circuits," in Proc. *Int. Symp. Fault-Tolerant Comput.*, Orlando, FL, pp. 44-49, June 1984.

[9] S. M. Reddy, M. K. Reddy and J. G. Kuhl, "On testable design for CMOS logic circuits," in Proc. *Int. Test Conf.*, Philadelphia, PA, pp. 435-445, Oct. 1983.

[10] N. K. Jha and J. A. Abraham, "Design of testable CMOS logic circuits under arbitrary delays," *IEEE Trans. CAD,* vol. CAD-4, no. 3, pp. 264-269, July 1985.

[11] S. M. Reddy and M. K. Reddy, "Testable realizations for FET stuck-open faults in CMOS combinational logic circuits," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 742-754, Aug. 1986.

[12] S. B. Akers, "Universal test sets for logic networks," *IEEE Trans. Comput.* vol. C-22, pp. 835-839, Sept. 1973.

[13] S. M. Reddy, "Complete test sets for logic networks," *IEEE Trans. Comput.*, vol. C-22, pp. 1016-1020, Nov. 1973.

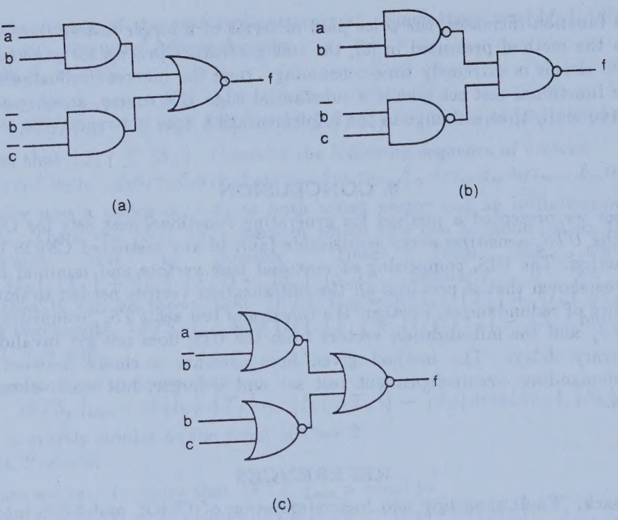

a
b
b
c
f
(a)
a
b
f
b
c
(b)
a
b
f
b
c
(c)

Figure 1. Restricted gate circuit implementations of $f = ab + \bar{b}c$

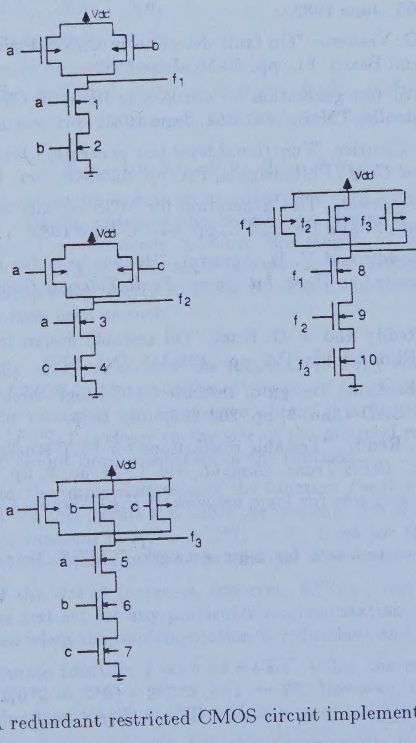

Vdd
a
b
f1
a
1
b
2
Vdd
a
c
f2
a
3
c
4
Vdd
f1
f2
f3
f
f1
8
f2
9
f3
10
Vdd
a
b
c
f3
a
5
b
6
c
7

Figure 2. A redundant restricted CMOS circuit implementation of $f = ab + ac$

ON THE DESIGN OF PARALLEL SCAN TESTABLE FINITE STATE SEQUENTIAL MACHINES

SUDARSHAN IYENGAR and RAMASWAMI DANDAPANI
Department of Electrical Engineering, University of Colorado at Colorado Springs, P. O. Box 7150, Colorado Springs, CO 80933-7150

SUDHAKAR M. REDDY
Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242

ABSTRACT

Classical scan designs require properly augmented flip-flops. In this paper a method to design scan testable sequential circuits using standard (i.e., unmodified) flip-flops is analyzed. In particular, performance degradation and area overhead needed to implement the scheme is studied for a number of sequential circuits realized using Programmable Logic Arrays.

I. INTRODUCTION

Designing digital circuits with testability considerations is of extreme importance in Very Large Scale Integrated (VLSI) technology. Several methods have been proposed for the design of testable synchronous sequential circuits. Among these, scan designs [1-4] are the most popular. In all these scan design methods the combinational part of the original sequential circuit is not modified, but the flip-flops are augmented and strung together into a shift register. Alternatively, it may be desirable to modify only the combinatorial part and not the system flip-flops. Such a scan design with standard flip-flops was proposed in [5]. The method does not require the details of realization or the state table of the sequential circuit. During testing, test patterns can be shifted in parallel into the circuit flip-flops, and the amount of parallelism can be controlled by the logic designer. The modifications are well-defined additions which can be incorporated into an existing design or it can become part of the design procedure for the combinational circuit part of the sequential circuit.

The combinational part of the sequential circuit can be designed as a highly structured circuit such as Programmable Logic Array (PLA) or Programmable Array Logic with buried flip-flops. A number of approaches to testing and testable design of PLAs have been proposed in recent years. Deterministic test sets are used in the testable designs of [6-8], whereas random testing is employed in the designs of [9-10]. The emphasis of these approaches is on PLAs realizing combinational functions. The techniques presented can be applied to sequential circuits realized using PLAs (called finite state machine PLA or FSM-PLA), but additional consideration has to be given to the testing of flip-flops in the feedback paths. In this paper, we analyze the design proposed in [5] and detailed in [11], for FSM-PLAs. The flip-flops in the design of [5] can be simply tested using the conventional shift test.

In Section II the design of [11] is briefly explained with respect to FSM-PLAs. The performance of a number of FSM-PLAs is given in Section III. Concluding remarks are contained in Section IV.

II. A SCAN TESTABLE DESIGN OF FSM-PLA

An FSM-PLA is shown in Figure 1. It has two primary inputs, three primary outputs, and four internal memory elements (flip-flops) which are not directly observable at primary outputs. There are nine product lines generating product terms P_1 through P_9. The functions generated at primary outputs are:

$O_1 = P_5 + P_7$,

$O_2 = P_1 + P_4 + P_8 + P_9$, and

$O_3 = P_2 + P_3 + P_7$.

Next state functions for the memory elements are:

$M_1 = P_1 + P_2 + P_3$,

$M_2 = P_1 + P_4 + P_5$,

$M_3 = P_1 + P_2 + P_5 + P_8$, and

$M_4 = P_5 + P_6$.

Classical scan design of the FSM-PLA shown in Figure 1 would require modification to the flip-flops to include (at least logically) a multiplexor at each flip-flop input to selectively accept scan-in data or normal function data [1]. The state of the FSM-PLA for a test is scanned in and the test is applied in one system clock cycle. The state information of the test result is scanned out. Note that a new test is serially shifted-in while the state resulting from the previous test is being shifted-out. When only one additional test mode pin is used, the test data is scanned in by "connecting" a primary input to the input stage of the scan path shift register and the output of the scan path shift register is multiplexed onto a primary output.

To ascribe the ability to shift a test pattern into the flip-flops and simultaneously shift-out the result of the previous test, a test mode input T and additional logic are added to the FSM-PLA being designed. The additional logic facilitates, when test mode input T = 1, writing into flip-flops from primary inputs as well as bringing the contents of the flip-flops out onto primary outputs. Simultaneously, the contents of the flip-flops which were not brought out are shifted into other flip-flops in a way that they can ultimately be brought out onto primary outputs. This is illustrated next through an example. Consider the PLA equations, given earlier, modified as below:

$O_1 = \bar{T}P_5 + \bar{T}P_7 + TM_4$

$O_2 = \bar{T}P_1 + \bar{T}P_4 + \bar{T}P_8 + \bar{T}P_9$

$O_3 = \bar{T}P_2 + \bar{T}P_3 + \bar{T}P_7$

$M_1 = \bar{T}P_1 + \bar{T}P_2 + \bar{T}P_3 + TI_1$

$M_2 = \bar{T}P_1 + \bar{T}P_4 + \bar{T}P_5 + TM_1$

$M_3 = \bar{T}P_1 + \bar{T}P_2 + \bar{T}P_5 + \bar{T}P_8 + TM_2$, and

$M_4 = \bar{T}P_5 + \bar{T}P_6 + TM_3$

As a result of these modifications, the machine operates normally when T = O and enters test mode when T = 1. In test mode, memory element M_1 is written into directly from I_1. Simultaneously, the previous values in M_1, M_2, M_3, and M_4 are shifted to M_2, M_3, M_4, and output O_1, respectively. These modifications introduce five added product terms, one for each of the added terms in the above equations. The modified FSM-PLA is given in Figure 2. The extra hardware needed is given by thin lines in Figure 2. It takes four clock cycles to shift-out the contents of flip-flops and simultaneously shift-in any desired state into the four flip-flops of the modified FSM-PLA.

Next consider the following modifications to the equations corresponding to the FSM-PLA of Figure 1:

$$O_1 = \bar{T}P_5 + \bar{T}P_7 + TM_3$$

$$O_2 = \bar{T}P_1 + \bar{T}P_4 + \bar{T}P_8 + \bar{T}P_9 + TM_4$$

$$O_3 = \bar{T}P_2 + \bar{T}P_3 + \bar{T}P_7$$

$$M_1 = \bar{T}P_1 + \bar{T}P_2 + \bar{T}P_3 + TI_1$$

$$M_2 = \bar{T}P_1 + \bar{T}P_4 + \bar{T}P_5 + TI_2$$

$$M_3 = \bar{T}P_1 + \bar{T}P_2 + \bar{T}P_5 + \bar{T}P_8 + TM_1, \text{ and}$$

$$M_4 = \bar{T}P_5 + \bar{T}P_6 + TM_2$$

As a result of these latter modifications, the machine operates normally when $T = O$ and enters test mode when $T = 1$. In test mode, memory elements M_1 and M_2 are written into directly from I_1 and I_2, respectively. Simultaneously, the previous values in M_1 and M_2 are shifted into M_3 and M_4, while the previous contents of M_3 and M_4 are shifted-out via primary outputs O_1 and O_2. These modifications introduce six additional product terms, one for each of the added terms in the above equations. The modified FSM-PLA is given in Figure 3. The product terms added are indicated by thin lines in Figure 3. It takes two clock cycles to shift-out the contents of flip-flops and simultaneously shift-in any desired state into the four flip-flops.

The above procedure can be generalized to any finite state machine. The number of shifts needed to set the memory elements to desired state depends upon the number of inputs, state variables and outputs of the original circuit as well as the design choices made as explained next. Let $\underline{m}$ be the number of flip-flops, $\underline{n}$ be the number of primary inputs, and $\underline{k}$ be the number of primary outputs of the original FSM-PLA. Then it can be readily seen that one can shift out the contents of up to min (k,m,n) flip-flops onto primary outputs, while simultaneously shifting in new values into the same number of flip-flops from primary inputs. That is, it is possible to "logically" connect the storage elements of FSM-PLA into a maximum of min (k,m,n) shift registers. If one chooses to shift R state variables, $R \leq$ min (k,m,n), then an additional (m + R) products will be needed in a PLA realization of the proposed testable FSM-PLA. The number of "shifts" or inputs needed to set the FSM-PLA into a desired state would be the least integer greater than or equal to m/R.

The design choices implied by the discussion given above do not affect the original PLA -- only new product lines and connections to the original product lines, to disable them when $T = 1$, are added. The additions made are decided solely on the number of primary inputs, outputs, internal memory elements and the choice of R defined above. No knowledge of the function realized or the implementation of the original PLA is needed to incorporate the design modifications. As a result, the final design may be reducible (a similar situation occurs in classical scan designs also).

The serial shift FSM-PLA obtained by using $R = 1$ is different from the classical scan designs, where the storage elements are modified. If the design procedure of [1] is applied to the FSM-PLA of Figure 1, then a modified FSM-PLA is obtained whose equations are:

$$O_1 = \overline{T}(P_5 + P_7) + TM_4$$

$$O_2 = P_1 + P_4 + P_8 + P_9$$

$$O_3 = P_2 + P_3 + P_7$$

$$M_1 = \overline{T}(P_1 + P_2 + P_3) + TI_1$$

$$M_2 = \overline{T}(P_1 + P_4 + P_5) + TM_1$$

$$M_3 = \overline{T}(P_1 + P_2 + P_5 + P_8) + TM_2, \text{ and}$$

$$M_4 = \overline{T}(P_5 + P_6) + TM_3$$

Thus, for classical scan designs it is necessary to modify the original FSM-PLA design by increasing the separation between input lines and output lines to accommodate the additions and/or modifications to storage elements.

We will analyze a number of FSM-PLAs for performance in the next section.

III. PERFORMANCE EVALUATION OF FSM-PLA

The parallel scan design given in [11] was analyzed for system performance with the aid of CAD software from the University of California at Berkeley. Six FSM-PLAs were created in nMOS technology. They are listed in Table 1. The scan design of these FSM-PLAs with $R = 1$, $R = \min(k,m,n)$ and $R =$ average between 1 and min (k,m,n) was performed and their design specifications are also listed in Table 1. These modified designs have been named with the suffixes -1, -MAX, -AVG, correspondingly. The six FSM-PLAs chosen have number of product terms ranging from 3 to 194. FSM-PLAs with larger number of product terms were not used due to system limitations. The data obtained does exhibit a trend that can be extrapolated to larger PLAs.

The parameters that were obtained for the four designs of each FSM-PLA are: (i) input to output propagation delay, (ii) area for layout, (iii) length and width for layout, (iv) number of transistors needed, and (v) capacitances at the output nodes. A single clock phase was used to drive the flip-flops. From a high level description of the FSM-PLA, the characteristics were obtained by a series of steps involving the programs PEG, EQNTOTT, MPLA, MEXTRA, CRYSTAL, and PLEASURE which are part of the 1986 VLSI tools package from the University of California at Berkeley. The programs were run on the Digital Equipment Corporation GPX VAX II station running ULTRIX 1.2.

The propagation delay is shown in Table 2. It lists the six FSM-PLAs and the percentage increase for each of the modifications. As seen from Figure 4 the percentage increase in delay shows a considerable drop with increasing product terms in all three modifications.

The percentage increase in area for the three modifications is given in Table 3. For small FSM-PLAs the modifications are area inefficient. As Figure 5 shows, the parallel scan designs require only about 12% increase in area overhead for larger PLAs.

Table 4 lists the changes to length and width due to the design modifications. As in the case of delay and area, these parameters generally decrease with increasing product terms.

Table 5 lists the percentage increase in transistors needed. The poly resistance was found constant at 30 Ω /sq. and the average capacitance at the output nodes varied from 0.003 pf to 0.008 pf which is low enough to cause minimum delay in subsequent stages.

IV. CONCLUSIONS

The parallel scan method of [11] was analyzed for performance. Three variations were studied which reflect the amount of parallelism in testing. The parameters area, delay and number of transistors all showed an increase for the three designs, but the percentage increase showed a downward trend for larger FSM-PLAs. The design modification can thus be considered as a viable alternative to testing the FSM-PLAs without appreciable drop in performance. The amount of parallelism to be introduced for enhancing testing speed can be determined from these results.

V. REFERENCES

1. M. J. Y. Williams and J. B. Angell, "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic," IEEE Trans. Comput., Vol. C-22, pp. 4-60, January 1973.

2. S. Funatsu, N. Wakatsuki and T. Arima, "Test Generation Systems in Japan," Proc. 12th Des. Automat. Symp., pp. 114-122, June 1975.

3. J. H. Stewart, "Future Testing of Large LSI Circuit Cards," Dig. 1977 Semiconductor Test Symp., 77CH1261-7C, pp. 6-17, October 1977.

4. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," J. Design Automat. Fault-Tolerant Comput., Vol. 2, pp. 165-178, May 1978.

5. S. M. Reddy and R. Dandapani, "Scan Design Using Standard Flip-Flops," IEEE Design & Test., pp. 52-54, February 1987.

6. K. A. Hua, J.-Y. Jou and J. A. Abraham, "Built-In Tests for VLSI Finite State Machines," Digest 14th Intl. Symp. Fault-Tolerant Comput., pp. 292-297, 1984.

7. H. Fujiwara, "A New PLA Design for Universal Testability," IEEE Trans. Comput., Vol. C-33, pp. 745-750, August 1984.

8. R. Treuer, H, Fujiwara and V. K. Agarwal, "Implementing a Built-In Self-Test PLA Design," IEEE Design & Test., pp. 37-48, April 1985.

9. E. M. Eichelberger and E. Lindbloom, "Random Pattern Coverage and Diagnosis fof LSSD Logic Self Test," IBM J. Res. and Develop., Vol. 27, pp. 265-272, May 1983.

10. D. S. Ha and S. M. Reddy, "On the Design of Random Pattern Testable PLAs," Proceedings Intl. Test. Conf., pp. 688-695, September 1986.

11. R. Dandapani and S. M. Reddy, "Parallel Scan Design Using Standard Flip-Flops for finite State Sequential Machines," accepted for presentation at the 6th Ann. IEEE Phoenix Conf. on Computers and Communications, March 1988.

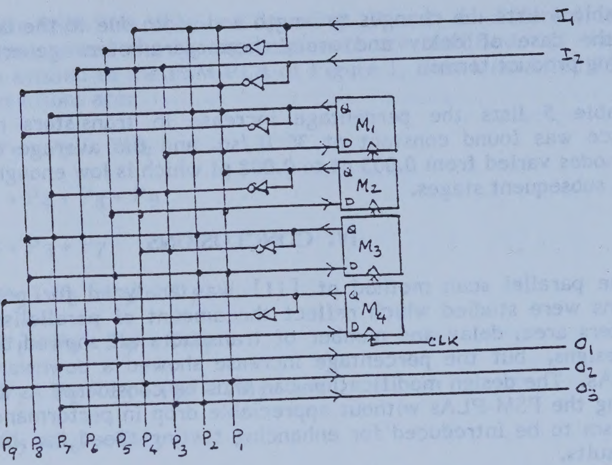

I1
I2
Q
M1
D
Q
M2
D
Q
M3
D
Q
M4
D
CLK
O1
O2
O3
P9 P8 P7 P6 P5 P4 P3 P2 P1

Figure 1. An FSM-PLA

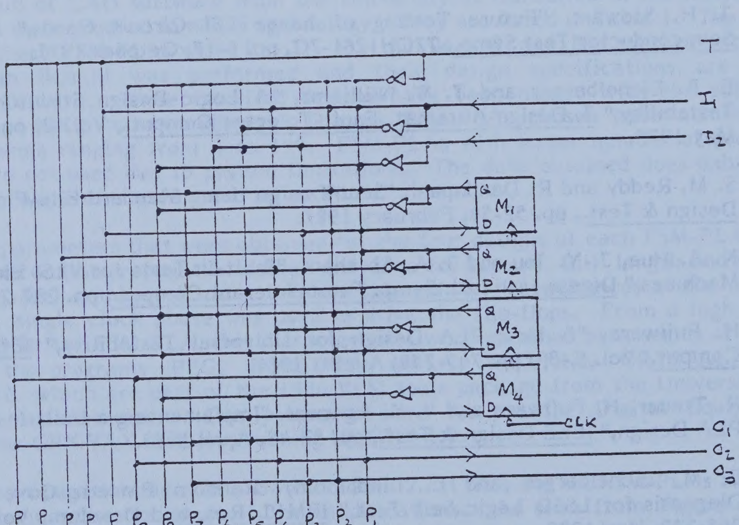

T
I1
I2
Q
M1
D
Q
M2
D
Q
M3
D
Q
M4
D
CLK
O1
O2
O3
P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1

Figure 2. FSM-PLA of Figure 1 modified for single shift register

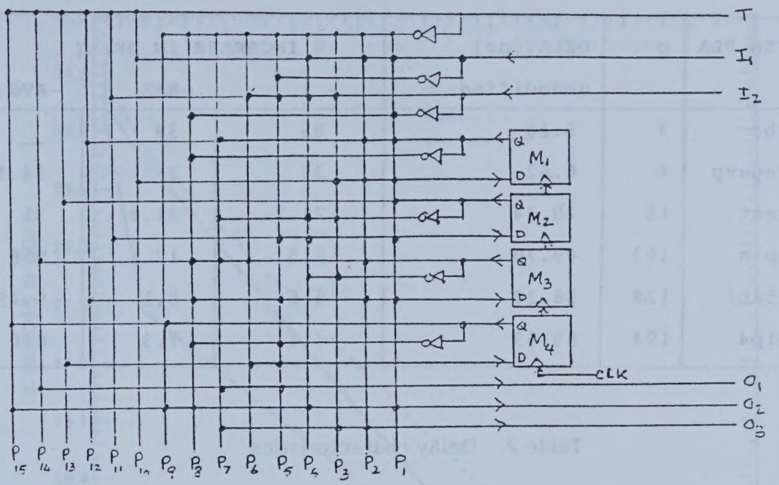

T
I1
I2
Q
M1
D
Q
M2
D
Q
M3
D
Q
M4
D
CLK
O1
O2
O3
P15
P14
P13
P12
P11
P10
P9
P8
P7
P6
P5
P4
P3
P2
P1

Figure 3. FSM-PLA of Figure 1 modified for two parallel shift registers

FSM_PLA	TYPE	n	k	p	m
2-bit binary counter	2bc	2	2	3	2
	2bc-1	3	2	6	2
	2bc-MAX	3	2	7	2
	2bc-AVG	-	-	-	-
(1/0)*100 detector	regexp	4	3	4	3
	regexp-1	5	3	8	3
	regexp-MAX	5	3	10	3
	regexp-AVG	5	3	9	3
error detector	test	4	7	15	3
	test-1	5	7	19	3
	test-MAX	5	7	21	3
	test-AVG	5	7	20	3
priority encoder	apla	10	12	103	12
	apla-1	11	12	114	10
	apla-MAX	11	12	123	10
	apla-AVG	11	12	118	10
Z=5*X+1 evaluator	z5xp1	7	10	128	5
	z5xp1-1	8	10	134	5
	z5xp1-MAX	8	10	138	5
	z5xp1-AVG	8	10	137	5
4-bit multiplier	mlp4	8	8	194	5
	mlp4-1	9	8	200	5
	mlp4-MAX	9	8	204	5
	mlp4-AVG	9	8	202	5

Table 1. FSM-PLAs and their design modifications

FSM_PLA	p	DELAY(ns)	% INCREASE IN DELAY		
		unmodified	- 1	-MAX	-AVG
2bc	3	5.16	35	39	—
regexp	4	6.42	33	36	34.1
test	15	10.34	21.2	31.6	31
apla	103	49.70	6.5	13.1	9.5
z5xp1	128	58.23	5.5	8.1	6.49
mlp4	194	85.53	4.1	7.3	6.0

Table 2. Delay characteristics

FSM_PLA	p	AREA (μ sqr)	% INCREASE IN AREA		
		unmodified	-1	-MAX	-AVG
2bc	3	2560	140	160	--
regexp	4	5760	86.6	110	100
test	15	22528	38.06	61.2	43.18
apla	103	280000	16.7	25.16	20.09
z5xp1	128	256256	12.5	16.8	13.1
mlp4	194	377728	10	16.0	12.4

Table 3. Area

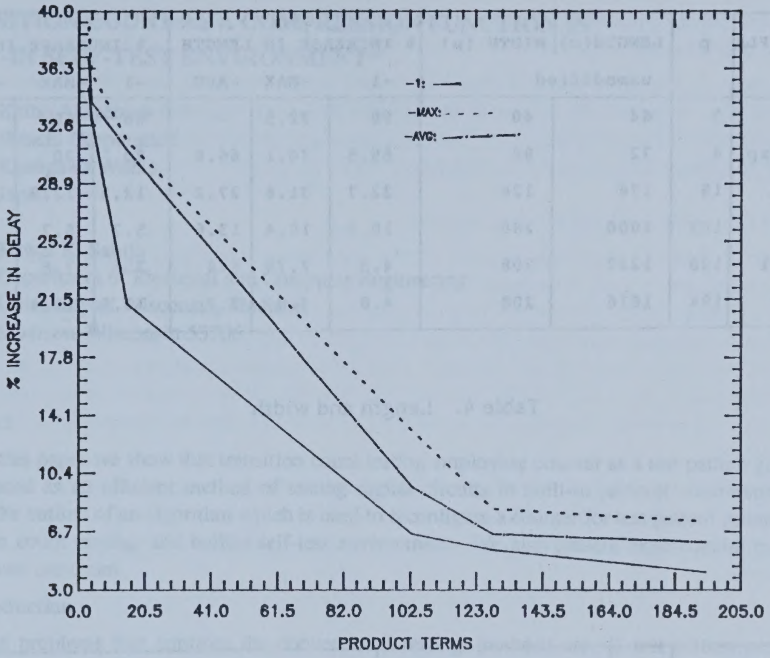


Figure 4. Product terms versus increase in delay

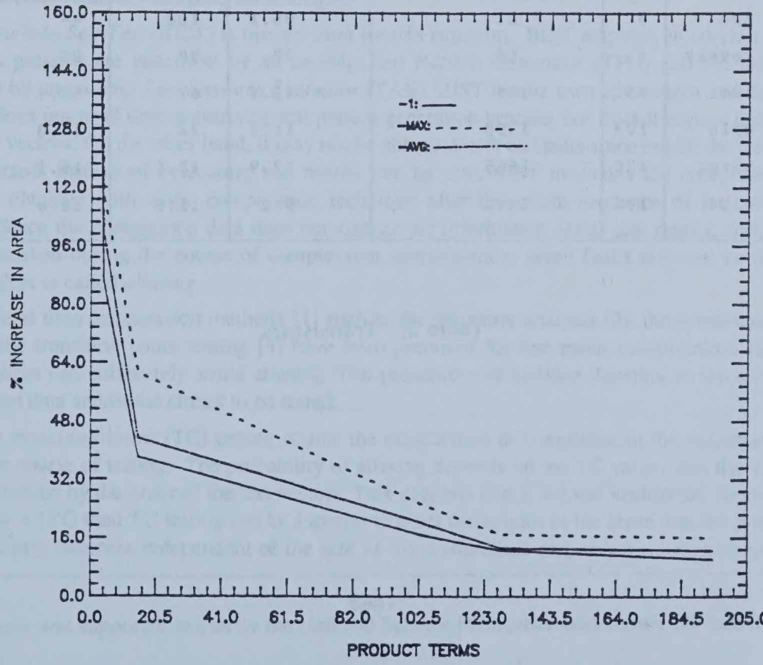


Figure 5. Product terms versus increase in area

FSM_PLA	p	LENGTH(µ)	WIDTH (µ)	% INCREASE IN LENGTH			% INCREASE IN WIDTH		
		unmodified		-1	-MAX	-AVG	-1	-MAX	-AVG
2bc	3	64	40	60	72.5	--	60	60	--
regexp	4	72	80	55.5	70.1	66.6	20	20	20
test	15	176	128	22.7	31.6	27.2	12.5	12.5	12.4
apla	103	1000	280	10.4	18.4	13.6	5.7	5.7	5.7
z5xp1	128	1232	208	4.5	7.79	6.4	7.6	7.6	7.6
mlp4	194	1816	208	4.8	7.0	5.7	11.5	11.5	11.5

Table 4. Length and width

FSM_PLA	p	# of TRANSISTORS	% INCREASE IN # OF TRANSISTORS		
		unmodified	-1	-MAX	-AVG
2bc	3	27	85.1	100	--
regexp	4	50	78	98	92
test	15	125	42.4	67	50.4
apla	103	1390	11.5	22	15.8
z5xp1	128	1665	12.9	15.6	15.1
mlp4	194	2342	9.2	11.5	10.8

Table 5. Transistors

TRANSITION COUNT AS A COMPRESSION FUNCTION IN BUILT-IN SELF-TEST ENVIRONMENT*

Keiho Akiyama#
Hitachi Corporation
Kanagawa Works
Japan

Kewal K. Saluja
Department of Electrical and Computer Engineering
University of Wisconsin, Madison
Madison, Wisconsin 53706

Abstract

In this paper we show that transition count testing employing counter as a test pattern generator can be used as an efficient method of testing digital circuits in built-in self-test environment. We present the outline of an algorithm which is used to reconfigure a counter for test pattern generation in transition count testing and builtin self-test environment. We also present experimental results to substantiate our claim.

1. Introduction

The problems that confront the conventional testing methods are: 1) test pattern generation takes excessive time, 2) the external tester is complicated and expensive, 3) it takes too mush time to apply tests, and 4) it is difficult to test and observe the entire circuits from outside with limited number of pins which are assigned for testing.

A *Built-In Self Test* (BIST) is one solution for this situation. BIST employs an on-chip test circuit. Test patterns are generated by an on-chip *Test Pattern Generator* (TPG) and the results are evaluated by an on-chip *Test Answere Evaluator* (TAE). BIST has its own advantages and disadvantages. It does not need time-consuming test pattern generation process, nor does it require the storage of all test vectors. On the other hand, it may not be able to detect all faults since not all the test results are evaluated. Instead of evaluating test results one by one, BIST evaluates the compressed data which is obtained with some compression technique after the entire sequence of test vectors is applied. Since the compressed data does not contain all information about test results, some errors may be masked during the course of compression, consequently, some faults may not be detected. This problem is called *aliasing*.

Several data compression methods [1] such as the signature analysis [2], the syndrome testing [3], and the transition count testing [4] have been proposed for test result compression. However, none of these can completely avoid aliasing. The probability of aliasing depends on the size of the compressed data and/or the circuit to be tested.

The *transition count* (TC) testing counts the occurrences of transitions in the output sequence during the course of testing. The probability of aliasing depends on the TC value. But the TC value can be cntrolled by the order of the test vectors. This suggests that if the test vectors are appropriately ordered by a TPG then TC testing can be superior to other techniques in the sense that the probability of the aliasing becomes independent of the size of the compressed data and the circuit to be tested.

---

* This work was supported in part by the National Science Foundation under grant DCR-8509194.

# The author was with the Department of Computer Sciences, University of Wisconsin-Madison.

Furthermore, if the probability of aliasing is made smaller, then TC testing can be considered more effective than other techniques.

In this paper, after reviewing the basic features of BIST, TC testing in BIST environment is discussed. Then, we present the outline of an algorithm to generate the configuration of a TPG which orders test vectors for TC testing such that aliasing occurs with a smaller probability compared to other methods. Following which the experimental results are given.

2. BIST Overview and Problem Formulation

BIST consists of three components: a TPG, the *Circuit Under Test* (CUT) which is usually combinational, and a TAE. The basic BIST structure is shown in the Figure 1. A control signal enables and disables the TPG and the TAE. In normal mode of operation the input signals from other regions of the chip go through the TPG and reach the CUT while the output signals go through TAE and reach their destinations. In the test mode, the CUT is separated from other parts of the chip and tested independently by using the TPG and the TAE. The TPG feeds a sequence of test vectors to the CUT. The TAE evaluates the responses of the CUT to these test vectors and reports the result to the outside world.

The simplest way to realize TPG and TAE is to use ROM But this can have only a limited practical application because of its high overhead. We shall restrict our attention to circuits using a TPG which applies exhaustive or pseudo exhaustive test patterns to the CUT and the TAE is a data compressor.

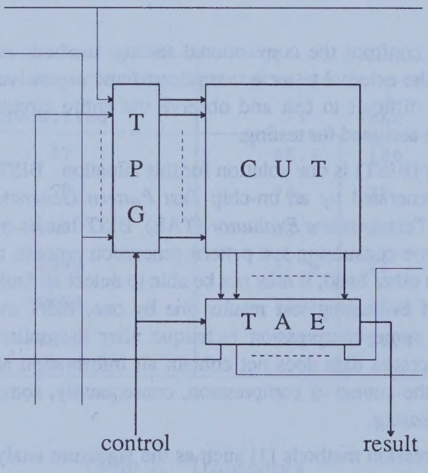


Figure 1: Basic BIST Configuration.

2.1. Test Pattern Generation

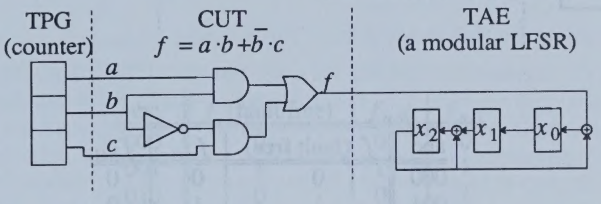
Exhaustive test patterns can be generated by counters, *Linear Feedback Shift Registers* (LFSR), etc.

2.2. Test Answer Evaluation

Data compression makes it possible to store the test results in a small space. The fundamental problem with the compression methods proposed in literature [1] is aliasing. How different compression methods deal with this problem is discussed below. For simplicity of presentation we initially assume that the CUT is a single output combinational circuit.

(1) Signature Analysis: In this method an n-bit binary sequence is compressed into r-bits by using an r-stage LFSR. The content of the LFSR after compression is called the *signature.*

The compression by signature analysis can be construed as a mapping from 2^n-space to 2^r-space. The aliasing occurs because it is possible that two distinct n-bit data are mapped onto the same signature. The probability of aliasing [5] is $(2^{n-r}-1)/(2^n-1) \approx 2^{-r}$ This shows that the probability of aliasing depends only on the length of the signature register. An example of this method and associated aliasing problem is shown in Figure 2.



abc	f (fault free)	contents of LFSR $x_2x_1x_0$	$f_{a/1}$	contents of LFSR $x_2x_1x_0$	$f_{f/1}$	contents of LFSR $x_2x_1x_1$
000	0	000	0	000	1	001
001	1	001	1	001	1	011
010	0	010	1	011	1	111
011	0	100	1	111	1	010
100	0	101	0	011	1	101
101	1	110	1	111	1	110
110	1	000	1	010	1	000
111	1	001	1	101	1	001
signature		001		101		001

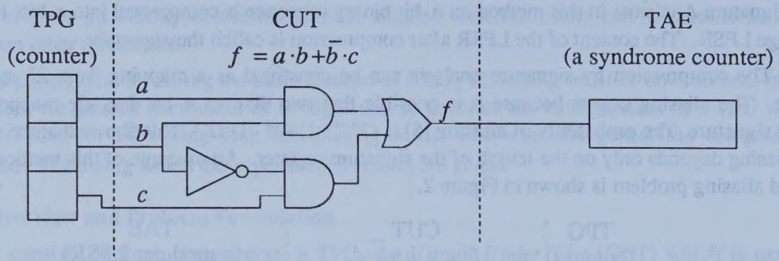
aliasing

Figure 2: An Example of Signature Analysis.

(2) Syndrome Testing: The syndrome value S for and r variable function with k minterms is defined as [3] $S = k/2^r$ After applying all possible 2^r input combinations to the CUT, an erroneous response is detected by comparing its syndrome value with the correct value. In the example shown in the Figure 3, the correct syndrome value is 4/8. The stuck-at 1 fault at the input a is detected because its syndrome value 6/8 is different from the correct value. On the other hand, a stuck-at 1 fault at the input b results into correct syndrome value, hence this fault is not detected.

By assuming that all errors occur with a constant probability, the effectiveness of the syndrome test can be computed. The probability of missing errors is $(\binom{2^r}{k}-1)/(2^{2^r}-1)$. This shows that larger the absolute value of $2^{r-1}-k$, the more errors are detected. Based on this observation Robinson and Saxena [6] defined *equivalent length, Le*, to compare the effectiveness of syndrome testing and signature analysis. *Le* is the length of an LFSR which misses as many errors as syndrome testing for a function with syndrome value $k/2^r$. Assuming that *Le* is continuous, it is shown in [6] that $Le = 2^r - \log_2 \binom{2^r}{k}$.

(3) Transition Count Testing: A TC value is the number of times the output value of the CUT changes during the course of the test. The TC test was not originally devised for BIST, but as a mean to make



abc	f (fault free)	$f_{a/1}$	$f_{b/1}$
000	0	0	0
001	1	1	0
010	0	1	0
011	0	1	0
100	0	0	1
101	1	1	1
110	1	1	1
111	1	1	1
k	4	6	4
S	4/8	6/8	4/8

aliasing

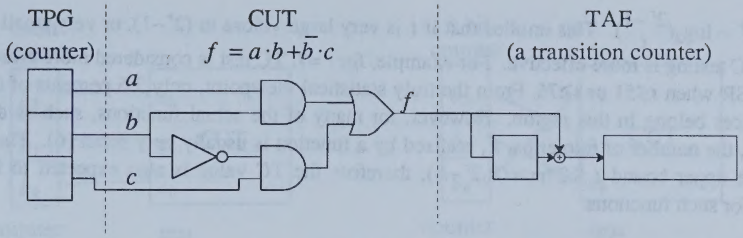
Figure 3: An Example of Syndrome Testing.

conventional testing efficient. Let t_i be a test vector whose fault-free response is r_i (r_i is either 0 or 1), and $T = t_1, t_2, \ldots, t_n$ be a set of test vectors. In the conventional test, all the fault-free responses, namely $R = r_1, r_2, \ldots, r_n$ are stored in the external memory. Faults are detected by comparing these with the actual responses. Let t_i^{α} be a test vector whose fault free response is α. The TC test sequence $S^* = t_p^0, t_1^0, t_2^0, \ldots, t_p^0, t_q^1, t_1^1, t_2^1, \ldots, t_q^1$, where $p + q = n$, detects all the faults detected by the conventional test if the circuit is realized as a two level circuit.

The TC testing concept can be used in BIST environment as shown in Figure 4. For the example circuit the fault-free TC value is 3. The input a stuck-at 0 fault is detected since the TC value is 4 in the presence of the fault. However, the fault a stuck-at 1 is not detected since its TC value is same as the fault free TC value. This is the aliasing problem for the TC test. It is shown in [6] that BIST with syndrome test and TC test together significantly improves the aliasing problem provided all erroneous sequences occur with equal probability.

2.3. Problem Formulation

The effectiveness of the TC testing depends only on the TC value if the equally likely occurrence of errors is assumed. However, unlike syndrome testing the TC value depends on the order of the input test vectors. This suggests that the TC testing is a useful TAE if test vectors are ordered in such a way that the aliasing probability in TC test is reduced. In the BIST environment with a counter or an LFSR as a TPG it may be impractical to arrange the test vectors in the sequence S^* shown above. But, by arranging the connection between the TPG and the CUT properly, the probability of aliasing can be reduced, thus making TC testing more effective TAE than signature analysis.



abc	f (fault free)	$f_{a/0}$	$f_{a/1}$
000	0	0	0
001	1	1	1
010	0	0	1
011	0	0	1
100	0	0	0
101	1	1	1
110	1	0	1
111	1	0	1
TC	3	4	3

aliasing

Figure 4: An Example of TC Testing.

In addition, if error-free TC value is very small, only a few counter bits are needed in the TAE. For example, for a 10 input circuit, the maximum possible TC value is 1023 requiring a 10 bit counter in the TAE. But if error-free TC value is small, say 25, then only a 5 bit counter is required in the TAE, while the effectiveness in terms of the probability of aliasing is still as good as for a 10 bit or a larger signature analyser.

Thus we need to investigate ways to implement a TPG such that the TC can be used as a more effective TAE then other compressors.

3. TC Testing in BIST Environment

As mentioned in the previous section, for the effective use of TC testing the test vectors must be generated in a certain order. To compare the effectiveness of the TC testing with the syndrome testing and the signature analysis we use the *equivalent length* concept for TC testing.

After discussing the basic characteristics of TC testing for single output circuits, the application to multiple output circuits will be discussed. As most of the actual logic units consist of multiple output circuits, the application of our method to multiple output circuits without losing its efficiency is an important issue if the method is to have any utilization in real environment.

3.1. Aliasing Problem in TC Testing

By assuming that all erroneous sequences are equally likely, the probability of aliasing is computed as follows. Let an r-input function f have the TC value t when applied an exhaustive test sequence (length 2^r) in a given order. The number of sequences with TC value t is $N(t) = \binom{2^r-1}{t} * 2$ Hence the probability of aliasing for the function f is $P(t) = (\binom{2^r-1}{t} * 2 - 1)/(2^{2^r} - 1)$ Using the arguments similar to Robinson and Saxena [6], we determine the equivalent length Le for TC testing as

$Le = 2^r - \log_2\binom{2^r-1}{t}$. This implies that if t is very large - close to (2^r-1), or very small - close to 0, then TC testing is more effective. For example, for r=7, TC test is considered more effective than 7-bit LFSR when $t \leq 51$ or $t \geq 76$. From the truly statistical viewpoint, only 3.6 percents of the length 2^7 sequences belong in this region. However, for many of the actual functions, such as decoders and PLA's, the number of minterms, k, realized by a function is usually very small [6]. The TC value t has the upper bound $t \leq 2*min(k, 2^r-k)$, therefore the TC value is also expected to be relatively small for such functions.

3.2. Arrangement of the Test Order in BIST Environment

It is evident that the bounds on the TC value are $1 \leq t \leq 2*min(k, 2^r-k)$. The minimum TC value, t=1, occurs when test vectors are ordered such that all tests whose expected responses are ones are applied prior to tests whose expected responses are zeros, or vice versa. We have also discussed that the aliasing probability in TC testing is reduced provided the TC value is very small or very large. Thus TC testing can be made more effective if the TC value by the chosen TPG is very small or very large. Furthermore, as mentioned in the previous section, the TC value is expected to be relatively small for most functions. Therefore, it seems reasonable to try to make TC value small by chosing a TPG appropriately.

For exhaustive testing of a r-input function there are $2^r!$ ways of ordering test vectors. The output sequence consists of k ones and (2^r-k) zeros. Therefore there are $\binom{2^r}{k}$ distinct sequences each of which occurs $k!(2^r-k)!$ times. If all k ones are either at the head or the tail of the sequence, TC value is one. This occurs in $2*(k!(2^r-k)!)$ ways.

In the BIST environment however, where the TPG is implemented by an LFSR or a counter, the desired ordering may not be possible but a rearrangement of the order of test vectors is possible by using the following methods (Figure 5): 1) *permutation* arranges the connections between the outputs of a TPG with inputs of a CUT, 2) *complementation* makes use of either uncomplemented or complemented outputs of a TPG, 3) *initialization* changes the starting test vector by initializing TPG as desired. The combination of these three methods, called *arrangements*, generate a variety of test orderings.

Although only a small fraction of all possible orders are possible using arrangements, yet the experimental results show that in most cases near optimum solution (very small TC value) can be obtained from these small choices [7].

Implementation of arrangements can be done with only a little overhead. For example, for permutation no extra circuit is needed since it merely changes the wirings. Similarly, complementary outputs can be extracted without extra inverters because each memory element of the counter is realized as a flip-flop. Initialization can be changed without extra cost.

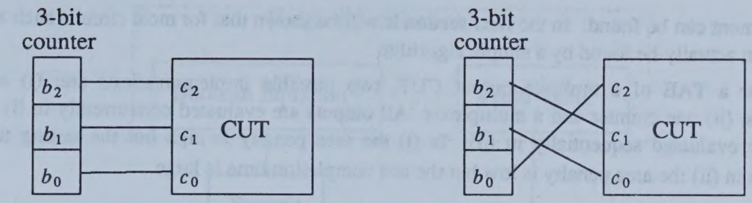
A TPG can be either a LFSR or a counter. If a counter is chosen as a TPG the upper bound on the value of TC by using the permutation and complementation methods are given in [7].

3.3. TC Testing for Multiple Output Circuits

A CUT with m $(m>1)$ outputs can be considered either as a single output circuit with all the outputs of the CUT processed through a space compressor [8] or as m single output circuits.

In the first case all m outputs are exclusive-ORed (a linear space compressor [8]) and the output of the space compressor is fed to the TAE (transition counter). The TPG is arranged such that the TC value of the exclusive-ORed output is small. While this idea is simple, the cost of the exclusive-OR tree can be prohibitive.

In the second case there are several ways to handle the problem. Three possible ways of implementation are to use: (i) m separate TPG's with multiplexers where each TPG corresponds to each output, (ii) one reconfigurable TPG which is able to modify the order of test vectors such that the

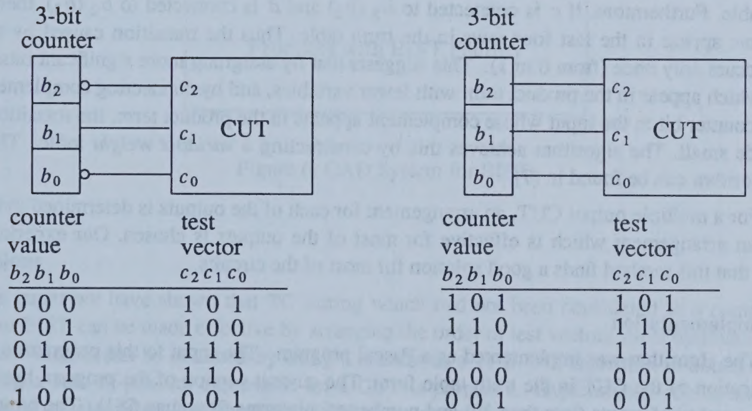

3-bit counter
b_2
b_1
b_0
c_2
c_1
CUT
c_0
3-bit counter
b_2
b_1
b_0
c_2
c_1
CUT
c_0

counter value $b_2 b_1 b_0$	test vector $c_2 c_1 c_0$
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 0 1
1 1 0	1 1 0
1 1 1	1 1 1

(no arrangement)

counter value $b_2 b_1 b_0$	test vector $c_2 c_1 c_0$
0 0 0	0 0 0
0 0 1	1 0 0
0 1 0	0 0 1
0 1 1	1 0 1
1 0 0	0 1 0
1 0 1	1 1 0
1 1 0	0 1 1
1 1 1	1 1 1

(1) permutation


3-bit counter
b_2
b_1
b_0
c_2
c_1
CUT
c_0
3-bit counter
b_2
b_1
b_0
c_2
c_1
CUT
c_0

counter value $b_2 b_1 b_0$	test vector $c_2 c_1 c_0$
0 0 0	1 0 1
0 0 1	1 0 0
0 1 0	1 1 1
0 1 1	1 1 0
1 0 0	0 0 1
1 0 1	0 0 0
1 1 0	0 1 1
1 1 1	0 1 0

(2) complementation

counter value $b_2 b_1 b_0$	test vector $c_2 c_1 c_0$
1 0 1	1 0 1
1 1 0	1 1 0
1 1 1	1 1 1
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0

(3) intialization

Figure 5: *Arrangemnt* of the Order of Test Vectors.

aliasing probability is small for each output, and (iii) one TPG which is shared among all outputs and is also effective for all of them. Among these, the first method is complete but costs too much, the second method may be implemented with smaller overhead than the first but the control logic might be very complicated to change the configuration as desired, the third method can be used without any area overhead. Therefore, the third method seems a good candidate for investigation if a satisfactory arrangement can be found. In the next section it will be shown that for most circuits such an arrangement can actually be found by a simple algorithm.

For a TAE of a multiple output CUT, two possible implementations are: (i) m separate counters, (ii) one counter and a multiplexer. All outputs are evaluated concurrently in (i) while outputs are evaluated sequentially in (ii). In (i) the area penalty is high but the testing time is low whereas in (ii) the area penalty is low but the test completion time is large.

4. An Algorithm to Generate TC TPG

In this section, a heuristic approach to make the TC testing more effective in BIST environment is proposed. To arrive at an appropriate arrangement, the algorithm make use of sum-of-product expression for the CUT. The algorithm is implemented as a program. In the algorithm it is assumed that a counter is used as an exhaustive TPG. Thus the goal is to find an arrangement for ordering the test vectors such that TC value for the CUT is as small as possible.

The basic strategy for arriving at a good arrangement is as follows. In order to make the TC value small, the rows of the truth table are to be ordered such that ones in the output columns appear in consecutive rows. Suppose an r-input function f contains a product term with i variables. This product term covers 2^{r-i} minterms. By connecting all i variables in the product term to the i most significant bits of the counter all 2^{r-i} minterms corresponding to the product term can be made to appear in the consecutive rows in the truth table. For instance, for the four input function $f = \bar{c}d + ab\bar{c} + \bar{a}bc\bar{d}$, the product term $\bar{c}d$ covers four minterms. if c and d are connected to b_3 and b_2 or vice versa then all four minterms covered by $\bar{c}d$ will appear in the four consecutive rows in the truth table. Furthermore, if c is connected to $\bar{b}_3$ ($\bar{b}_2$) and d is connected to b_2 (b_3), then those four minterms appear in the last four rows in the truth table. Thus the transition caused by this product term occurs only once (from 0 to 1). This suggests that by assigning more significant bits to the variables which appear in the product term with fewer variables, and by connecting complemented output of the counter bit to the input whose complement appears in the product term, the transition count can be made small. The algorithm achieves this by constructing a *variable weight table*. The details of the algorithm can be found in [7].

For a multiple output CUT, an arrangement for each of the outputs is determined independently. Then an arrangement which is effective for most of the outputs is chosen. Our experimental study shows that this method finds a good solution for most of the circuits.

4.1. Implementation

The algorithm was implemented as a Pascal program. The input to this program is a functional specification of the CUT in the truth table form. The current version of the program has a restriction on the number of inputs (less than 17) and number of minterms (less than 581). The program is interfaced with the Berkeley CAD tools, It can be used as a subsystem in the CAD system. The system configuration including this program is shown in the Figure 6.

5. Experimental Results

As an experiment 32 PLAs are considered for application of the method. The summary of the results is given in Table 1. The results are compared with the signature analysis using an r-bit signature analyser where r is the number of inputs to the CUT. For 91.3% of the 240 outputs, the proposed method gives better results than the signature analysis. The proposed method needs 20 percent fewer number of bits in the TAE. The comparison of the effectiveness of the method in terms of equivalent length *Le* is even more impressive.

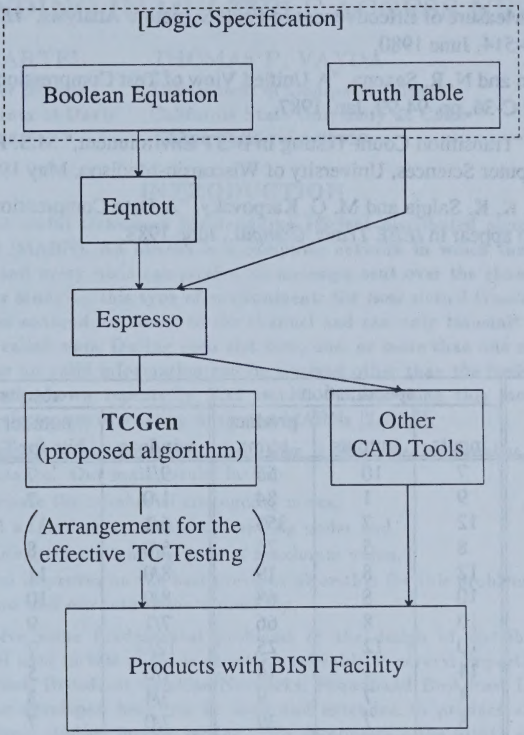

[Logic Specification]
Boolean Equation
Truth Table
Eqntott
Espresso
TCGen
(proposed algorithm)
Other
CAD Tools
Arrangement for the
effective TC Testing
Products with BIST Facility

Figure 6: CAD System for BIST.

6. Conclusions

In this paper we have shown that TC testing which had not been considered as a compaction technique for BIST can be made effective by arranging the order of test vectors. Near optimum ordering of the test vectors can be obtained by using a counter as a TPG. An algorithm to obtain a good solution using a sum-of-product expression for a CUT is proposed. Experimental results show the effectiveness of the algorithm. For more than 90% of the cases, the algorithm finds arrangements that make TC testing more effective than the signature analysis.

References

[1] J. S. Upadhyaya and Kewal K. Saluja, "Signature Techniques in Fault Detection and Location," in *Spectral Techniques in Fault Detection*, Editor M. Karpovsky, Academic Press, Orlando, 1986.

[2] R. A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," *Hewlett-Packard J.*, pp. 2-8, May 1977.

[3] J. Savir, "Syndrome-Testable Design of Combinational Circuits," *IEEE Trans. Comput.*, vol. C-29, pp. 442-451, June 1980.

[4] J. P. Hayes, "Transition Count Testing of Combinational Logic Circuits," *IEEE Trans. Comput.*, vol. C-25, pp. 613-620, June 1976.

[5] J. E. Smith, "Measure of Effectiveness of Fault Signature Analysis," *IEEE Trans. Comput.*, vol. C-29, pp. 510-514, June 1980.

[6] J. P. Robinson and N. R. Saxena, "A Unified View of Test Compression Methods," *IEEE Trans. Comput.*, vol. C-36, pp. 94-99, Jan. 1987.

[7] K. Akiyama, "Transitiion Count Testing in BIST Environment," M.S. Research Report, Department of Computer Sciences, University of Wisconsin-Madison, May 1987.

[8] S. M. Reddy, K. K. Saluja and M. G. Karpovsky, "A Data Compression Technique for Built-In Self-Test," To appear in *IEEE Trans. Comput.*, June 1988.

PLA	specification			results		
	inputs	outputs	product terms	win/lose	number of bits for TAE	Le
z5xp1	7	10	65	9/1	7	65.9
z9sym	9	1	84	1/0	7	151.9
add6	12	7	355	6/1	11	1834.9
adr4	8	5	75	5/0	8	104.5
alu1	12	8	19	8/0	11	1767.3
alu2	10	8	68	8/0	10	408.6
alu3	10	8	66	7/1	9	611.1
apla	10	12	25	12/0	7	778.4
co14	14	1	14	1/0	5	16107.4
dc1	4	7	9	3/4	3	3.9
dc2	8	7	39	7/0	7	55.2
dist	8	5	120	5/0	8	103.8
dk17	10	11	18	11/0	7	806.5
dk27	9	9	10	9/0	5	419.5
dk48	15	17	21	17/0	11	28539.7
f51m	8	8	76	7/1	8	143.1
gary	15	11	107	11/0	9	31229.1
in0	15	11	107	11/0	9	31229.1
in1	16	17	104	17/0	10	62157.1
mlp4	8	8	127	5/3	7	72.4
pla	8	8	127	5/3	7	72.4
radd	8	5	75	4/1	7	78.4
rd53	5	3	31	3/0	5	6.4
rd73	7	3	127	3/0	7	13.2
risc	8	31	28	31/0	4	233.6
root	8	5	57	5/0	4	218.6
sqn	7	3	38	3/0	6	28.7
sqr6	6	12	50	8/4	6	19.1
temp	5	3	31	3/0	5	6.4
tial	14	8	579	8/0	13	5394.6
wim	4	7	9	4/3	3	5.0
z4	7	4	59	3/1	7	35.5
average	9.1	8.2	85.0	7.5/0.7	7.2	5523.7

Table 1: Experimental Results.

MAXIMUM FINDING IN MULTIPLE ACCESS NETWORKS

CHARLES U. MARTEL
Division of Computer Science
University of California at Davis
Davis, California, 95616

THOMAS P. VAYDA
Department of Computer Science
California State University at Chico
Chico, California, 95929

INTRODUCTION

We consider several useful techniques for designing efficient distributed algorithms in a *multiple access broadcast network* (MABN). An MABN is a computer network in which there is a single shared communication channel and every node can receive all messages sent over the channel. We use a commonly accepted model for studying this type of environment: the *time slotted broadcast channel.* In this model, the network nodes *contend* for access to the channel and can only transmit during discrete synchronized time intervals called *slots.* During each slot zero, one, or more than one nodes can transmit a message. In the third case no valid information can be received other than the feedback that a *collision* has occurred. It has been shown repeatedly that results derived using this model provide a good approximation for the performance of existing unslotted MABNs [2,3,6,7].

The performance metric used throughout the paper is the *expected* number of slots required to solve a given problem instance. Our main results include:

- a fast algorithm to estimate the number of contending nodes,
- a fast algorithm to find a small subset of the contending nodes and
- a fast algorithm for finding the contending node of maximum value.

In each case our algorithm improves on the best previous algorithm for this problem. In the next section we will define the problems and our results more precisely.

Our algorithms solve some fundamental problems in the design of distributed algorithms for MABNs. Since the model used in this paper is directly applicable to several important existing network environments (e.g. Ethernet, Broadcast Satellite Networks, Broadband Broadcast LANs, etc.), it is our hope that the algorithms developed here can be used and extended to produce efficient solutions for many of the other problems studied in this setting. We demonstrate the utility of our algorithms by obtaining improved bounds for several important MABN problems.

MAIN RESULTS

In this section we give a precise definition of the problems and a summary of our results. We will concentrate on the *maximum finding problem on values from a known distribution.* This problem can be described as follows: Let C be any subset of the network nodes such that $|C| \equiv Q > 1$. We will refer to a node in C as a *contending node.* Every node in C has a numeric value drawn from a known continuous distribution. (i.e. The node values are IID random variables.) The problem is to find the largest of the contending nodes' values while using as few slots as possible [4]. Another problem that we will discuss briefly is the *selection resolution problem.* Here, each of the contending nodes tries to broadcast during the same slot and this results in a collision. A selection resolution algorithm attempts to grant one of the contending nodes access to the channel while minimizing the number of slots used.

We will consistently use the following notation: N is the *total* number of network nodes, Q is the number of *contending* nodes in a particular invocation of an algorithm and q^* is an *estimate* for Q.

Our first result is a distributed probabilistic algorithm that rapidly finds an estimate q^* for Q. Efficient algorithms for *estimating* the number of contending nodes have many applications, and a number of estimation methods have been developed [2,3,5]. The first well known method for estimation/selection resolution, *truncated binary exponential backoff,* is currently used on the standard Ethernet [1]. This estimation method uses $O(logN)$ expected slots, while our estimator runs in $O(loglogN)$ slots.

We determine q^* by a probabilistic binary search over the powers of two between 2 and $2^{\lceil \log N \rceil}$. The estimator uses a probabilistic operation to test the current value of the estimate q^*. During each slot, every contending node transmits with probability $1/q^*$; if this results in a collision then q^* is probably too small and if no nodes transmit then q^* is probably too big. In this way, the feedback can be used to determine the next value of q^*. This technique is a straightforward extension of the super-exponential binary search which Willard used to develop an $O(loglogN)$ expected time selection resolution algorithm in [7]. The performance of our estimator is summarized by the following theorem.

THEOREM 1. The estimation procedure has the following properties:
i) It outputs an estimate $q^* \in Z$ where $Z = \{ 2^1, 2^2, \ldots, 2^{\lceil logN \rceil} \}$.
ii) At most $2(\lfloor loglogN \rfloor + 1)$ slots are required to obtain the estimate q^*.

$$\text{iii) } P(q^* = q) \leq \begin{cases} 4(Q/q)^2 & \text{for } 0 < Q/q \leq 1 \\ e^{-Q/q} & \text{for } 1 \leq Q/q < \infty \end{cases} \quad \text{for all} \quad q \in Z.$$

The bounds given in part iii) are useful since they both approach zero rather quickly as Q/q deviates away from 1. We can loosely interpret this to mean that the output q^* from our estimator is likely to be a good estimate for Q. Later we show that this estimate is of sufficient quality to improve the best current bounds for some significant broadcast network problems.

One way to speed up a given algorithm is to do some "pre-processing" which eliminates many of the items. We call such a pre-processing method a *reducer*. A reducer should *quickly* produce a *significant* reduction in the problem size. Our second algorithm is a distributed probabilistic reducer that uses the nodes' IID values to rapidly isolate a *small* subset of contenders. It uses an estimate q^* for Q to produce an estimate v^* for the largest of the contending nodes' values. In each slot, all nodes with values greater than v^* transmit. If no nodes transmitted, v^* is reduced and this is repeated until the first non-empty slot. The nodes that broadcast in the final slot comprise the reduced set of contenders. The quality of a reducer can be measured by its running time and the number of nodes it eliminates. Both of these clearly depend on the accuracy of the estimate q^*. An analysis of our reducer shows that

THEOREM 2. The reducer, using the estimate q^* produced by our estimator, reduces the problem size to an expected *constant* number of nodes in an expected *constant* number of slots.

This technique solves the maximum finding problem in less than 2.526... expected slots when Q is known in the *first* slot. Our algorithm is computationally less expensive and simpler than previously known methods, yet our bound improves on most of the bounds obtained in [6].

Our main result is an ***exponential*** improvement over the previous best time of $O(logN)$ expected slots for the maximum finding problem when Q is unknown and the node values are drawn from a known distribution. The maximum finder is a distributed probabilistic algorithm which has three phases. It first uses the estimator to obtain an estimate q^* for Q, then it uses the reducer to eliminate most of the contending nodes and then finds the maximum of the R remaining nodes' values in $O(logR)$ expected slots using a maximum finding algorithm due to Wah and Juang [6].

THEOREM 3. Our maximum finder locates the maximum in $2 \lfloor \mathbf{loglogN} \rfloor + O(1)$ expected slots.

The algorithms can be combined in various ways to produce other efficient algorithms for MABNs. For example, we can show that the estimator can be combined with the reducer to improve the expected time of the Ethernet selection resolution algorithm from $O(logN)$ to $O(loglogN)$ slots, which matches the best algorithm for this problem [7]. A complete description of these results is given in [4].

ACKNOWLEDGMENT

This work was supported by NSF Grant MCS-83-11422 and by IBM and the State of California through project MICRO.

REFERENCES

[1] Digital Equipment, Intel and Xerox Corps. *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specification, Version 2.0.* Xerox Corp., Palo Alto, Calif., November 1982.

[2] Greenberg, A. *Efficient Algorithms for Multiple Access Channels.* TR 83-08-01, Department of Computer Science, University of Washington, Seattle, Washington 98195, August 1983.

[3] Greenberg, A., Flajolet, P. and Ladner, R. Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels. *Journal of the ACM,* Vol. **34**, No. 2, April 1987.

[4] Martel, C. U. and Vayda, T. *Load Estimation, Problem Size Reduction and Maximum Finding in Multiple-Access Broadcast Networks,* Technical Report CSRL-87-3, Computer Science Research Laboratory, University of California at Davis, Davis, Ca., 95616, May 1987.

[5] Metcalfe, R. and Boggs, D. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM,* Vol. **19**, No. 7, July 1976

[6] Wah, B. and Juang, J. Resource Scheduling for Local Computer Systems with a Multiaccess Network. *IEEE Transactions on Computers,* Vol. **C-34**, No. 12, December 1985.

[7] Willard, D. E. Log-Logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM Journal of Computing,* Vol. **15**, No. 2, May 1986.

COLLISION RESOLUTION ALGORITHMS FOR A TIME-CONSTRAINED MULTIACCESS CHANNEL[1]

SHIVENDRA S. PANWAR,[2] DON TOWSLEY[3],
JACK K. WOLF[4], YEHUDA ARMONI [2]

Abstract

Collision Resolution Algorithms (CRA's) for the ternary feedback multiple access channel with time constraints are considered in this paper. We wish to maximize the number of packets that are successfully transmitted within a fixed deadline K after their arrival for transmission. Packet arrivals are assumed to be Poisson. A non-nested CRA is described and it's performance is compared with a nested CRA for values of $K \leq 4$.

1. INTRODUCTION

The problem of a large number of occasionally active users attempting to share a common channel is referred to as the multiaccess communications problem [1-3]. Whenever two or more users attempt to transmit a packet at the same time a collision results. When this happens a collision resolution algorithm (CRA) is used to control retransmissions so that they are eventually successfully transmitted. The objective of these algorithms usually have been to maximize the throughput of the channel while maintaining a finite average delay. A different objective is necessary when considering time-constrained traffic such as packetized voice and distributed sensor networks. For such time constrained traffic a packet has to be transmitted within a certain time, called its deadline, or is considered useless and is lost. In such an environment it is desirable to devise CRA's which minimize the packet loss for a given arrival rate. In this paper we devise a non-nested CRA for a time-constrained multiaccess channel. We also compare the performance of the non-nested CRA with a nested CRA.

2. MODEL

A standard model for study in the field of multiple access communications has been the slotted channel with ternary feedback [1]. The channel is slotted and each slot corresponds to the length of a packet. Ternary feedback is made available to all the users. That is, at the end of each slot, every user is told whether the slot was idle, had a successful transmission, or experienced a collision between two or more packets. The distribution of the sum traffic generated by all the users is assumed to be Poisson. Several CRA's have been proposed for this channel [e.g. 1,2]. We will assume, in addition, that all packets have identical deadlines from their time of arrival at the users for transmission. If the packet does not successfully commence transmission within it's deadline it is considered lost. Kurose et al [3] have studied a time-constrained CSMA/CD channel and computed the loss for different time constraints. A window algorithm similar to the one used in [1] is employed and is modeled as the server of an M/G/1 queue with customer losses.

---

[1] This work was supported by the Center for Advanced Technology in Telecommunications, Polytechnic University, Brooklyn, NY 11201.

[2] Department of Electrical Engineering, Polytechnic University,333 Jay Street, Brooklyn, NY 11201.
[3] Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003.
[4] Center for Magnetic Recording Research, Univerity of California at San Diego, La Jolla, CA 92093.

3. DESCRIPTION OF CRA

Let all the packet arrivals have a deadline of K time units, where a time unit corresponds to the length of a slot. This deadline is the maximum allowable delay from the packet's time of arrival to the commencement of it's successful transmission. Packets which are not transmitted within this deadline are lost. Let the total arrival rate be λ per unit time it and the percentage loss be $L(\lambda, K)$. We wish to devise CRA's that minimize $L(\lambda, K)$. Let the current time at the beginning of a slot be t_{CT}. Then a window CRA need only enable intervals in the time interval $[t_{CT} - K, t_{CT}]$, since arrivals before $t_{CT} - K$ will already have been lost. For such a model it is easy to determine the optimal CRA for $K \leq 1$.

3.1 The optimal CRA for K ≤ 1. Consider the arrival time axis just before the beginning of the k^{th} slot $(k = 1, 2, 3, ...)$. As mentioned earlier, we are only concerned with arrivals in the time interval $(t_k - K, t_k)$, where t_k is the current time at the beginning of the k^{th} slot. At the end of the k^{th} slot, the part of the arrival axis that should be considered for the next (i.e. $(k+1)^{th}$)) slot is $(t_{k+1} - K, t_{k+1})$ or, since $t_{k+1} = t_k + 1, (t_k + 1 - K, t_k + 1)$. Since $K \leq 1, (t_k - K, t_k)$ and $(t_k + 1 - K, t_k + 1)$ are disjoint time intervals, each of length K. Therefore, the enabling of any portion of the time interval $(t_k - K, t_k)$ for transmission in the k^{th} slot cannot affect or give any information about the time interval $(t_{k+1} - K, t_{k+1})$. Consequently, the random process of whether a success, idle or collision is observed in successive slots is independent of it's past history, no matter which CRA is employed. Therefore, in order to minimize the loss it is sufficient to maximize the probability of success in the next slot. at the beginning of each slot we have available a time interval of length K time units with Poisson distributed arrivals. In order to maximize the probability of success in the next slot, it caneasily be shown that we should enable an interval of length $E(\lambda, K)$ given by

$$E(\lambda, K) = \begin{cases} K, & \text{if } \lambda \leq 1/K \\ 1/\lambda, & \text{if } \lambda > 1/K. \end{cases} \tag{1}$$

The corresponding throughput $TP(\lambda, K)$ is given by

$$TP(\lambda, K) = \begin{cases} \lambda K e^{-\lambda K}, & \text{if } \lambda \leq 1/K \\ e^{-1}, & \text{if } \lambda > 1/K \end{cases} \tag{2}$$

and the minimum loss $L(\lambda, K)$ is given by

$$L(\lambda, K) = \begin{cases} \lambda - \lambda K e^{-\lambda K}, & \text{if } \lambda \leq 1/K \\ \lambda - e^{-1}, & \text{if } \lambda > 1/K. \end{cases} \tag{3}$$

3.2 CRAs for K > 1. CRAs for the ternary feedback channel with $K > 1$ will be considered next. For values of $K > 1$ the optimal CRA could not be determined. Instead a CRA with the first-come, first-served (FCFS) policy is devised. This CRA ensures that packets are transmitted in the order of generation. In order to compute the performance of the CRAs we use the value iteration method [4]. This method is briefly summarized next.

Consider a finite state, discrete-time, ergodic Markov chain. After each transition, the system is in one of N states $i, i = 1, 2, ..., N$. For each state i, an action $k = 1, 2, ..., K_i$ is chosen. Then p_{ij}^k is the probability of transition to state j if the process is in state i and action k was chosen. Associated with each transition from i to j, under action k, is a reward r_{ij}^k. Let the value function v_i^n be the total expected reward from the next n transitions, if the system is now in state i, and if an optimal policy is followed. The expected value can be written as

$$v_i^{n+1} = \max_{1 \leq k \leq K_i} \left[\sum_{j=1}^{N} p_{ij}^k \{r_{ij}^k + v_j^n\}\right], i = 1, 2, ..., N. \tag{4}$$

Let g be the maximal expected gain per transition. The principal goal in Markov decision processes is to compute g and to find the policy that achieves g. This can be done using the following theorem due to Odoni [6].

Theorem: Let all stationary policies have transition probability matrices representing single-chain aperiodic Markovian processes. Define x_i^n by

$$x_i^n = v_i^{n+1} - v_i^n. \tag{5}$$

Then for any choice of boundary conditions v_i^0,

1) $x_i^n \to g, 1 \le i \le N$.
2) $L_{max}^n = \max\limits_i x_i^n$ is monotonically decreasing in n to g.
3) $L_{min}^n = \min\limits_i x_i^n$ is monotonically increasing in n to g.
4) Any policy π achieving the N maxima in (4) for all n greater than or equal to some n_0 has the maximal gain per transition.

The value iteration algorithm proceeds as follows. The value functions $v_i^n, n = 1, 2, ...,$ are computed using equation (4). This computation is carried out until $L_{max}^n - L_{min}^n < \epsilon$, where ϵ is the maximum allowable error in the computation of g. Then $g \approx 1/2[L_{max}^n + L_{min}^n]$, since $L_{min}^n \le x_i^n \le L_{max}^n$ from the theorem. The optimal policy which achieves the maximal gain g is the one which achieves the N maxima in (4) for all $n \ge n_0$.

We applied this method to determine the operation of the CRA and to compute the losses. In a manner similar to that described in [1,2], three types of states are generated. However, due to the presence of time constraints, the relative position of the window with respect to current time (t_{CT}) has to be included in the state description. The optimal initial window size and the splitting of the collision intervals is also dependent on how far the window is behind current time. An additional complication is that part of the window may be " left behind," i.e. a part or an entire window under the process of being resolved may fall more than K time units behind current time. As a result the arrivals in this window are lost. The three classes of states are described next (also see Fig.1).

$S_0[h]$: In this state there are no untransmitted arrivals in the time interval $(t_k - K, t_k - h)$, while nothing more than the a priori information is known about the time interval $(t_k - h, t_k), 0 \le h \le K$.

$S_1[h, m, k]$: In this state there are no untransmitted arrivals in the time interval $(t_k - K, t_k - h)$. The time interval $(t_k - h, t_k - h + m)$ is part of an interval of length $(m + k)$ in which we know that there is at least one arrival, and of which k time units are more than K time units in the past, $0 \le h \le K, 0 \le m \le K - h, 0 \le k \le K - h - m$.

$S_2[h, m, k]$: In this state there are no untransmitted arrivals in the time interval $(t_k - K, t_k - h)$. The time interval $(t_k - h, t_k - h + m)$ is part of an interval of length $(m + k)$ in which we know that there are at least two arrivals, and of which k time units are more than K time units in the past, $0 \le h \le K, 0 \le m \le K - h, 0 \le k \le K - h - m$.

The value iteration equations corresponding to these three sets of states are listed in the appendix. The reward is set to 1 for a successful transmission and 0 for a collision or an idle slot. As a result the maximal gain per transition corresponds to the throughput. $VS_0^n[h], VS_1^n[h, m, k]$ and $VS_2^n[h, m, k]$ are the expected rewards corresponding to states $S_0[h]$, $S_1[h, m, k]$ and $S_2[h, m, k]$, respectively, in the next n steps if the optimal policy is followed. The event of observing an idle, success or collision if an interval of length j is enabled is represented by $q_j = 0, q_j = 1$ or $q_j \ge 2$, respectively.

The CRA determined by equations A.1 - A.3 is a non-nested CRA. This is due to the fact that we allow enabled sets that are not subsets of busy sets for states $S_1[h, m, k]$ and $S_2[h, m, k]$ when

$m \leq h-(K-1)$ and $K-1 < h \leq K$. For this set of states it is allowable to enable a part of the Poisson distributed interval of length $h-(K-1)-m$ along with the interval of length m corresponding to the part of the busy interval with packets that have not exceeded their deadlines. Since this interval will be lost by the next slot and lead to a transition to the $S_0[K]$ state in any case, this does not introduce new states. This option was introduced as it may lead to a higher probability of success. If this option is not allowed in equations A.2 and A.3, they would correspond to a nested CRA as described in [5].

In order to make the state space finite the time axis is discretized by dividing each time unit into sufficiently small "minislots". Losses were computed for both the nested and non-nested CRA's. Due to the fact that the state space increases rapidly with K, the performance of these CRA's could only be computed for values of $K \leq 4$. The loss for the non-nested CRA is plotted as a function of K for a range of values of λ in Figure 2. As expected, the losses decrease with K for a fixed value of λ. The slope of the loss curve showed an interesting discontinuity for integer values of K. The non-nested variation of the CRA described above had lower losses than the nested CRA for values of $1 < K \leq 2$ (see Table 1). The performance of the nested and non-nested CRA's is identical for values of K outside this range. We believe, however, that other non-nested CRA's can lead to a further reduction in the loss.

4. CONCLUSIONS

We considered the problem of devising CRA's for a time-constrained multiple access channel with ternary feedback. Poisson arrivals and a fixed deadline for all packets was assumed. The optimal CRA for the case when the time constraint was less than one packet transmission time was determined. We devised two variations of the window CRA, one nested and the other non-nested, for the case when the time constraint exceeded one packet transmission time. The value iteration method was used to determine the optimal enabled time intervals and the packet losses corresponding to different packet arrival rates and time constraints. The non-nested CRA was shown to lead to lower losses for a range of values of the time constraint.

REFERENCES

[1] R.G. Gallager, "Conflict resolution in random access broadcast networks," in *Proc. AFOSR Workshop Commun. Theory and Applications*, Provincetown, MA, Sept. 1978, pp.74-76.

[2] J. Mosely and P.A. Humblet, "A class of efficient contention resolution algorithms for multiple access," *IEEE Trans. Commun.*, vol. COM-33, pp. 145-152, Feb. 1985.

[3] J. Kurose, M. Schwartz and Y. Yemini," Multiple-Access Protocols and Time Constrained Communication," *ACM Computing Surveys*, vol. 16, no. 1, March 1984.

[4] R. Howard, *Dynamic Programming and Markov Processes*, Cambridge, MA, MIT Press, 1960.

[5] S.S. Panwar, "Time Constrained and Multiaccess Communications", Ph.D. Dissertation, Electrical and Computer Eng. Dept., University of Massachusetts, Amherst, MA, Feb. 1986.

[6] A. R. Odoni, "On Finding the Maximal Gain for Markov Decision Processes", *Operations Research*, Vol. 17, No. 5, Sept.-Oct. 1969.

APPENDIX

$$VS_0^{n+1}[h] = \left\{ \begin{array}{ll}
\max\limits_{0<j\le h} & \{P(q_j=0)VS_0^n[h-j+1]+ \\
 & P(q_j=1)\{VS_0^n[h-j+1]+1\} \\
 & +\,P(q_j\ge 2)VS_0^n[h+1,j,0]\}\,, \text{for } h\le K-1 \\
\max & \Big[\max\limits_{0<j\le h-(K-1)}\{P(q_j=1)+VS_0^n[K]\}, \\
 & \max\limits_{h-(K-1)<j\le h}\{P(q_j=0)VS_0^n[h-j+1] \\
 & +P(q_j=1)\{VS_0^n[h-j+1]+1\}+ \\
 & P(q_j\ge 2)VS_2^n[K,K-1-h+j,h-K+1]\}\Big], \\
 & \text{for } K-1<h\le K.
\end{array}\right. \tag{A.1}$$

$$VS_1^{n+1}[h,m,r] = \left\{ \begin{array}{ll}
\max\limits_{0<j\le m} & [P(q_j=0|q_{m+r}\ge 1)VS_1^n[h-j+1,m-j,r]+ \\
 & +P(q_j=1|q_{m+r}\ge 1)\{VS_0^n[h-j+1]+1\} \\
 & +P(q_j\ge 2|q_{m+r}\ge 1)VS_2^n[h+1,j,0]]\,, \text{for } h\le K-1 \\
\max & \Big[\max\limits_{0<j\le m}\{P(q_j=1|q_{m+r}\ge 1)+VS_0^n[K]\}\,, \\
 & \max\limits_{m<j\le h-(K-1)}\{P(q_j=1|q_{m+r}\ge 1)P(q_{j-m}=0) \\
 & +P(q_j=0|q_{m+r}\ge 1)P(q_{j-m}=1)+VS_0^n[K]\}\Big], \\
 & \text{for } m\le h-(K-1), K-1<h\le K. \\
\max & \Big[\max\limits_{0<j\le h-(K-1)}\{P(q_j=1|q_{m+r}\ge 1)\{VS_0^n[K]+1\}+ \\
 & P(q_j=0|q_{m+r}\ge 1)VS_0^n[K,m-h+K-1,r+h-K+1-j] \\
 & +P(q_j\ge 2|q_{m+r}\ge 1)VS_0^n[K]\}, \\
 & \max\limits_{h-(K-1)<j\le m}\{P(q_j=0|q_{m+r}\ge 1)VS_1^n[h-j+1,m-j,r]+ \\
 & +P(q_j=1|q_{m+r}\ge 1)\{VS_0^n[h-j+1]+1\}+ \\
 & P(q_j\ge 2|q_{m+r}\ge 1)VS_2^n[K,K-1+j-h,h-K+1]\}\Big], \\
 & \text{for } h-(K-1)\le m\le h, K-1<h\le K
\end{array}\right. \tag{A.2}$$

$$VS_2^{n+1}[h,m,r] = \left\{ \begin{array}{ll}
\max\limits_{0<j<m} & [P(q_j=0|q_{m+r}\ge 2)VS_2^n[h-j+1,m-j,r]+ \\
 & +P(q_j=1|q_{m+r}\ge 2)\{VS_1^n[h-j+1]+1\} \\
 & +P(q_j\ge 2|q_{m+r}\ge 2)VS_2^n[h+1,j,0]]\,, \text{for } h\le K-1 \\
\max & \Big[\max\limits_{0\le j\le m}\{P(q_j=1|q_{m+r}\ge 2)+VS_0^n[K]\}\,, \\
 & \max\limits_{m<j\le h-(K-1)}\{P(q_j=1|q_{m+r}\ge 2)P(q_{j-m}=0) \\
 & +P(q_j=0|q_{m+r}\ge 2)P(q_{j-m}=1)+VS_0^n[K]\}\Big], \\
 & \text{for } m\le h-(K-1), K-1<h\le K. \\
\max & \Big[\max\limits_{0<j\le h-(K-1)}\{P(q_j\ge 2|q_{m+r}\ge 2)VS_0^n[K]+ \\
 & P(q_j=0|q_{m+r}\ge 2)VS_2^n[K,m-h+K-1,r+h-K+1-j]+ \\
 & P(q_j=1|q_{m+r}\ge 2)\{VS_1^n[K,m-h+K-1,r+h-K+1-j]+1\}\}, \\
 & \max\limits_{h-(K-1)<j\le m}\{P(q_j=0|q_{m+r}\ge 2)VS_2^n[h-j+1,m-j,r]+ \\
 & +P(q_j=1|q_{m+r}\ge 2)\{VS_1^n[h-j+1,m-j,r]+1\}+ \\
 & P(q_j\ge 2|q_{m+r}\ge 2)VS_2^n[K,K-1+j-h,h-K+1]\}\Big], \\
 & \text{for } h-(K-1)\le m\le h, K-1<h\le K
\end{array}\right. \tag{A.3}$$

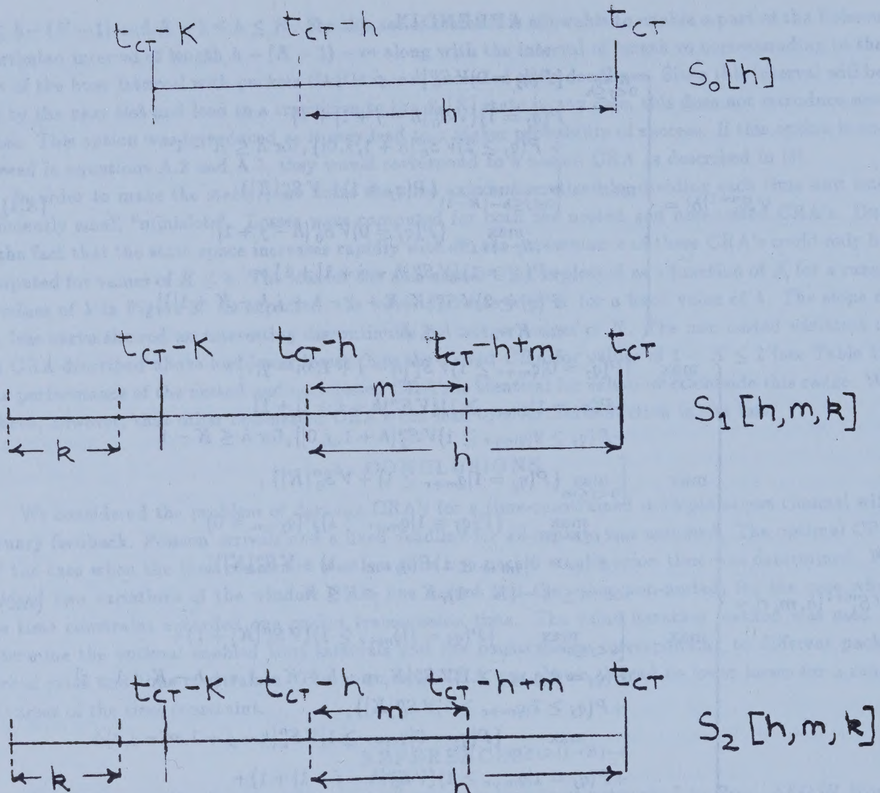

t_CT-K
t_CT-h
t_CT
S_0[h]
h
t_CT-K
t_CT-h
t_CT-h+m
t_CT
m
S_1[h,m,k]
k
h
t_CT-K
t_CT-h
t_CT-h+m
t_CT
m
S_2[h,m,k]
k
h

FIGURE 1. Definition of states.

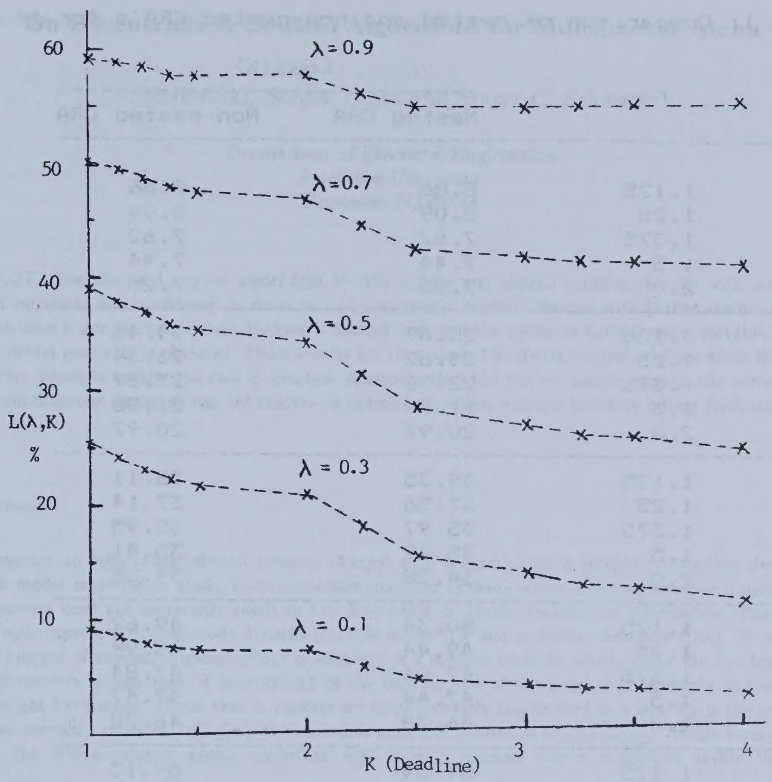


FIGURE 2. Loss vs Deadline for non-nested CRA

TABLE 1. Comparison of nested and non-nested CRA's for $1 < K \leq 2$

λ	K	Loss(%)	
		Nested CRA	Non-nested CRA
0.1	1.125	8.86	8.66
	1.25	8.09	8.04
	1.375	7.62	7.62
	1.5	7.44	7.44
	2.0	7.25	7.25
0.3	1.125	25.67	24.46
	1.25	23.62	23.34
	1.375	22.38	22.37
	1.5	21.89	21.85
	2.0	20.97	20.97
0.5	1.125	39.35	38.11
	1.25	37.56	37.14
	1.375	35.97	35.95
	1.5	35.42	35.41
	2.0	34.28	34.28
0.7	1.125	50.34	49.62
	1.25	49.44	48.94
	1.375	47.86	47.84
	1.5	47.49	47.47
	2.0	46.78	46.78
0.9	1.125	59.34	59.13
	1.25	59.18	58.66
	1.375	57.84	57.70
	1.5	57.74	57.71
	2.0	57.66	57.66

On Decentralized Control Algorithms for Multipacket Aloha

Sylvie Ghez, Sergio Verdú and Stuart C. Schwartz

Department of Electrical Engineering
Princeton University
Princeton NJ 08544

ABSTRACT. Decentralized control algorithms for the infinite-user slotted collision channel with multipacket reception capability are considered. In order to find the optimal control, the case with perfect state information where the users know the value of the backlog is studied first, and the optimum throughput achievable by decentralized control protocols is obtained. Those results are then applied to derive control schemes when the backlog is unknown, which is usually the case in practice. Assuming that the receiver cannot demodulate correctly more than K simultaneous transmissions, we analyze an optimal algorithm which is based on binary feedback.

1. The model

Consider an infinite-user slotted collision channel with the multipacket reception capability described in [4]. This model is useful to study multiuser communication systems where the simultaneous transmission of several packets does not necessarily result in the destruction of all the transmitted information. These include systems with capture ([5], [10]), code division multiple access [13] and multiuser detectors ([16]). It is assumed that the number of successful transmissions in each slot is a random variable, which, given the number of packets simultaneously transmitted, is independent of the backlog and of the number of attempts at transmission packets might have made. Given that n packets are simultaneously transmitted in a slot, ϵ_{nk} is the probability that k are correctly received $(0 \leq k \leq n)$; the reception matrix is defined as $(\epsilon_{nk})_{n \geq 1, 0 \leq k \leq n}$. It has been proved in [4] that the Aloha random access algorithm with such a channel has a maximum stable throughput $\eta_0 = \lim_{n \to \infty} C_n$, where $C_n = \sum_{k=1}^{n} k \epsilon_{nk}$ is the average number of packets correctly received in collisions of size n, and the limit η_0 is assumed to exist.

Decentralized control strategies have been shown ([6], [7], [14]) to stabilize the slotted Aloha algorithm with the usual collision channel, and in this paper we study the maximum stable throughput achievable by those strategies in the multipacket reception channel. Let us consider schemes of the form

$$\begin{aligned} p_n &= F(S_n) \\ S_{n+1} &= G(S_n, Z_n) \end{aligned} \tag{1}$$

where p_n is the retransmission probability in slot n, S_n is an estimate of the backlog X_n at the beginning of slot n, and Z_n is the feedback at the end of slot n. The number of new packets arriving during slot n, $(A_n)_{n \geq 0}$, is assumed to be a sequence of i.i.d. random variables with probability distribution $P(A_n = k) = \lambda_k \quad (k \geq 0)$, such that the mean arrival rate $\lambda = \sum_{n=1}^{\infty} n \lambda_n$ is finite. Each of the A_{n-1} new packets that arrived during slot $n-1$ is transmitted in slot n with probability p_n.

As in the case of conventional collision channels, it is useful to study first the case of control with perfect state information where the value of the backlog is known prior to the selection of the retransmission probability. Even though this situation is not relevant in practical systems, its study, which is carried out in Section 2, is motivated by the fact that in all random acces systems analyzed so far, the throughput achievable with channel feedback equals that achievable with perfect state information. In Section 3, we drop the assumption on perfect state information and restrict our attention to multipacket channels where the number of successful transmissions per slot cannot exceed some fixed integer K. The estimate of the backlog studied in Section 3 is recursively updated using a binary feedback which indicates only whether each slot was empty or not. It is shown that the throughput achievable with this type of feedback is the same as the perfect state information throughput.

This research was partially supported by the Office of Naval Research under Contract N00014-87-k-0054 and by the Army Research Office under Contract DAAL03-87-k-0062.

2. Control of multipacket Aloha with perfect state information

In this section we let the retransmission probability be a function of the exact value of the backlog at the beginning of slot n, i.e., $p_n = F(X_n)$. In this ideal case, the system is much simpler to analyze, since $(X_n)_{n\geq 0}$ is a homogeneous Markov chain. Our goal is to determine the optimal control F^* that maximizes the stability region, and the corresponding throughput η_c. For instance, it is well known [3] that for the usual collision channel with the delayed first transmission rule that we are using here, $F^*(X_n) = 1/X_n$ is the optimal retransmission probability, resulting in an ideal throughput of $\eta_c = e^{-1}$. In reality the users do not know the value of the backlog, but the results of this section give an upper bound which is actually tight on the maximum stable throughput of any control scheme of the form (1). They also suggest the use of certainty-equivalence retransmission probability of the form $p_n = F^*(S_n)$ in the partial state information case.

We first determine the throughput of the system if we simply set $p_n = A/X_n$, which is analogous to the optimal retransmission probability for the usual collision channel, and then we show that this is in fact optimal, provided that the constant A is properly chosen.

Lemma 1. If A is any positive constant and if $p_n = A/X_n$ for $X_n \geq A$, then the system is stable for $\lambda < t(A)$ and unstable for $\lambda > t(A)$ with $t(A) = e^{-A} \sum_{n=1}^{\infty} C_n \frac{A^n}{n!}$.

Proof. $(X_n)_{n\geq 0}$ is a homogeneous Markov chain which evolves according to

$$X_{n+1} = X_n + A_n - \Sigma_n \tag{2}$$

where Σ_n is the number of packets successfully transmitted in slot n. The system is defined to be stable if $(X_n)_{n\geq 0}$ is ergodic and unstable otherwise. Let d_i be the drift of X_n at state i: $d_i = E[\,X_{n+1} - X_n \mid X_n = i\,]$. We have $0 \leq \Sigma_n \leq X_n$, and for $i \geq \max\{1, A\}$

$$P[\,\Sigma_n = k \mid X_n = i\,] = \sum_{j=k}^{i} \binom{i}{j} \left(\frac{A}{i}\right)^j \left(1-\frac{A}{i}\right)^{i-j} \epsilon_{jk} \qquad (0 \leq k \leq i)$$

It then follows from (2) that for $i \geq \max\{1, A\}$

$$d_i = \lambda - \sum_{k=1}^{i} k \sum_{j=k}^{i} \binom{i}{j} \left(\frac{A}{i}\right)^j \left(1-\frac{A}{i}\right)^{i-j} \epsilon_{jk} = \lambda - \sum_{j=1}^{i} \binom{i}{j} \left(\frac{A}{i}\right)^j \left(1-\frac{A}{i}\right)^{i-j} C_j \tag{3}$$

We now show that the drift limit is $\lambda - t(A)$. We have for $i \geq M+1$:

$$\begin{aligned} |\,d_i - (\lambda - t(A))\,| \leq{}& \sum_{j=1}^{M} \left| \frac{i(i-1)\cdots(i-j+1)}{i^j} \left(1-\frac{A}{i}\right)^{i-j} - e^{-A} \right| \frac{A^j}{j!} C_j \\ &+ \sum_{j=M+1}^{i} \frac{i(i-1)\cdots(i-j+1)}{i^j} \left(1-\frac{A}{i}\right)^{i-j} \frac{A^j}{j!} C_j + e^{-A} \sum_{j=M+1}^{\infty} \frac{A^j}{j!} C_j \end{aligned}$$

Let T_1, T_2 and T_3 be the three terms on the right hand side of this equation, and ϵ any positive real. Since $T_2 + T_3 \leq 2 \sum_{j=M+1}^{\infty} \frac{A^j}{j!} C_j$, we can fix M such that $T_2 + T_3 < \epsilon$. Then each of the M terms in T_1 goes to zero when i goes to infinity. Therefore $\lim_{i\to\infty} d_i = \lambda - t(A)$. Since it is clear from (3) that the drifts are bounded, it follows from Pakes Lemma [11] that $(X_n)_{n\geq 0}$ is ergodic if $\lambda < t(A)$. It is shown in Appendix A that Kaplan's condition holds and therefore if $\lambda > t(A)$ Kaplan's result [8] applies and $(X_n)_{n\geq 0}$ is nonergodic. Note that the value of p_n for $X_n < A$ is left unspecified because it does not affect the throughput.

If $\lim_{n\to\infty} C_n = 0$, the control $p_n = A/X_n$ stabilizes the system. But for some reception matrices, there may not exist a constant A such that $t(A) > \eta_0$, i.e., no throughput improvement can be obtained with a retransmission probability which is inversely proportional to the backlog. In particular, if $C_n \geq \eta_0$ for all $n \geq 1$, (for instance, this is the case for a system with perfect capture and no noise, or also for the model with mobile users and pairwise transmissions developed in [4]), it can be proved with simple drift analysis considerations that no decentralized control protocol of the form (1) can achieve a throughput larger than η_0. In the remainder of this section, we will assume that the sequence $(C_n)_{n\geq 1}$ is nonincreasing and nonconstant, which is true in most cases of interest. Then there does exist an A such that $t(A) > \eta_0$ and furthermore it is easy to see that $t(A)$ is maximized by the unique solution A^* of the equation

$$C_1 = \sum_{n=1}^{\infty} A^n \frac{C_n - C_{n+1}}{n!} \tag{4}$$

It follows from the theorem below that this simple control scheme is optimal.

Theorem 1. If $(C_n)_{n\geq 1}$ is nonincreasing and nonconstant, there exists a unique retransmission probability $p_n^* = F^*(X_n)$ minimizing the drift d_i at state i. With the control $p_n^* = F^*(X_n)$, the system is stable for $\lambda < \eta_c$ and unstable for $\lambda > \eta_c$, with $\eta_c = e^{-y} \sum_{n=1}^{\infty} C_n \frac{y^n}{n!}$ where y is the unique solution of $C_1 = \sum_{n=1}^{\infty} y^n \frac{C_n - C_{n+1}}{n!}$.

Proof. If $X_n = i \geq 1$ and if we denote by p the retransmission probability used in slot n, we have

$$d_i(p) = \lambda - \sum_{j=1}^{i} \binom{i}{j} p^j (1-p)^{i-j} C_j \tag{5}$$

and

$$\frac{1}{i} d_i'(p) = \sum_{j=1}^{i-1} \binom{i-1}{j} p^j (1-p)^{i-j-1} (C_j - C_{j+1}) - (1-p)^{i-1} C_1 \tag{6}$$

so that if $p \neq 1$, $d_i'(p) = 0$ iff

$$\sum_{j=1}^{i-1} \binom{i-1}{j} x^j (C_j - C_{j+1}) = C_1 \tag{7}$$

where we have defined $x = \frac{p}{1-p}$. Since $(C_n)_{n\geq 1}$ is nonincreasing and nonconstant, $C_1 > 0$ and the left hand side of (7) is a strictly increasing function of x with range $[0, +\infty[$. Therefore (7) has a unique solution x_i^* corresponding to a unique zero of $d_i'(p)$, $p_i^* = \frac{x_i^*}{1+x_i^*}$. If $p=1$, then $d_i'(p) = 0$ iff $C_{i-1} = C_i$ but this corresponds to a local maxima.

Let us now study some properties of the sequence $(p_i^*)_{i\geq 1}$. We start by showing that $\lim_{i\to\infty} p_i^* = 0$, which is intuitively obvious. Since $p_i^* = \frac{x_i^*}{1+x_i^*}$, it is enough to prove that $x_i^* = \frac{p_i^*}{1-p_i^*}$ decreases to zero. Define $h_i(x) = \sum_{j=1}^{i-1} \binom{i-1}{j} (C_j - C_{j+1}) x^j$. Then from (7), x_i^* is the only solution of $h_i(x) = C_1$. It is easily checked that for all $x \geq 0$, $h_{i+1}(x) \geq h_i(x)$, and since both functions are strictly increasing, we can conclude that $(x_i^*)_{i\geq 1}$ is a decreasing sequence. Denoting by l the smallest integer such that $C_l - C_{l+1} > 0$, we get for $i > l+1$: $h_i(x) \geq \binom{i-1}{l} (C_l - C_{l+1}) x^l$, so that $C_1 \geq \binom{i-1}{l} (C_l - C_{l+1}) (x_i^*)^l$, which can be written $x_i^* \leq \frac{D}{(i-1)^{1/l}}$, where D is a constant. Therefore $\lim_{i\to\infty} x_i^* = 0$.

The next step of the proof is to show that $(ip_i^*)_{i\geq 1}$ has a limit y which is the only solution of $C_1 = \sum_{n=1}^{\infty} y^n \frac{C_n - C_{n+1}}{n!}$. Let us define $y_i = ix_i^* = \frac{ip_i^*}{1-p_i^*}$. Then (y_i) has a limit iff (ip_i^*) has one, and these limits are equal. From (7), y_i is the only solution of $g_i(z) = C_1$, where we have defined

$$g_i(z) = \sum_{j=1}^{i-1} \frac{(i-1) \cdots (i-j)}{i^j} \frac{C_j - C_{j+1}}{j!} z^j \tag{8}$$

It is easily shown from (8) that $g_{i+1}(z) \geq g_i(z)$. Since both functions are increasing, it follows that $y_{i+1} \leq y_i$ and therefore that $(y_i)_{i\geq 1}$ has a limit y. From (6), $d_i'(p) < 0$ for $p < \frac{1}{i}$, so we necessarily have $ip_i^* \geq 1$ and $y \geq 1$. Next we show that $\lim_{i\to\infty} g_i(y_i) = \sum_{n=1}^{\infty} y^n \frac{C_n - C_{n+1}}{n!}$, and therefore we will have $\sum_{n=1}^{\infty} y^n \frac{C_n - C_{n+1}}{n!} = C_1$. Define $\Delta_i = g_i(y_i) - \sum_{n=1}^{\infty} y^n \frac{C_n - C_{n+1}}{n!}$. Then for $M \leq i-2$:

$$|\Delta_i| \leq \sum_{j=1}^{M} \left| \frac{(i-1) \cdots (i-j)}{i^j} y_i^j - y^j \right| \frac{C_j - C_{j+1}}{j!} + 2 \sum_{j=M+1}^{\infty} y_1^j \frac{C_j - C_{j+1}}{j!}$$

M can be fixed so that the second term is smaller than any $\epsilon > 0$, and then each term in the first summation goes to zero as i goes to infinity.

Now we proceed to find the maximum stable throughput of the system if p_i^* is the retransmission probability, by computing again the drift limit, $\lim_{i\to\infty} d_i(p_i^*) = \lambda - e^{-y} \sum_{n=1}^{\infty} y^n \frac{C_n}{n!} = \lambda - \eta_c$. From (5), and with

$z_i = ip_i^*$:

$$d_i(p_i^*) = \lambda - \sum_{j=1}^{i} \frac{(i-1)\cdots(i-j+1)}{i^j}\, z_i^{\,j}\, (1-\frac{z_i}{i})^{i-j}\, \frac{C_j}{j!}$$

We have

$$| \, d_i(p_i^*) - \eta_c \, | \le \sum_{j=1}^{M} \left| \, \frac{i(i-1)\cdots(i-j+1)}{i^j}\, z_i^{\,j}\, (1-\frac{z_i}{i})^{i-j} - e^{-y}\, y^j \, \right| \frac{C_j}{j!}$$

$$+ \sum_{j=M+1}^{i} \frac{i(i-1)\cdots(i-j+1)}{i^j}\, z_i^{\,j}\, (1-\frac{z_i}{i})^{i-j}\, \frac{C_j}{j!} + e^{-y} \sum_{j=M+1}^{\infty} y^n\, \frac{C_n}{n!}$$

If y, $z_i \le Z$, the last two terms on the right hand side of this equation are smaller than $\sum_{j=M+1}^{\infty} Z^j \frac{C_j}{j!} < \epsilon$ for M large enough. Having fixed such an M, each term in the first sum goes to zero. It is shown in Appendix A that Kaplan's condition holds for this system; then the Theorem follows from the results in [8] and [11].

The main conclusion of this section is that it is not necessary to compute the exact value of $F^*(i)$, since we have $t(A^*) = \eta_c$, meaning that if we set $p_n = A^*/X_n$ for $X_n \ge A^*$, the maximum stable throughput of the system is η_c. It is intuitively obvious that no decentralized control algorithm of the form (1) can have a maximum stable throughput larger than η_c. This can indeed be proved by using a result of [15]. Consider the Markov chain (X_t, S_t), and the Lyapunov function $V(n,s)=n$. Assume that $\lambda > \eta_c$. Then

$$E[V(X_{t+1}, S_{t+1}) - V(X_t, S_t) \mid X_t=n,\, S_t=s] = \lambda - \sum_{j=1}^{n} \binom{n}{j} F(s)^j\, (1-F(s))^{n-j}\, C_j \ge d_n(p_n^*) \ge \frac{\lambda-\eta_c}{2}$$

for all n large enough and all s. Therefore the drift of V is strictly positive outside a finite subset of the state space. Since it is shown in Appendix A that the generalized Kaplan's condition is verified, it is enough to conclude that (X_t, S_t) is nonergodic.

Several examples are gathered in Table 1 below.

	C_n	$\eta_0 = \lim_{n\to\infty} C_n$	$\eta_c = \sup_{A>0} e^{-A} \sum_{n=1}^{\infty} C_n \frac{A^n}{n!}$
conventional collision channel [1]	$1 \quad n=1$ $0 \quad n>1$	0	e^{-1}
q-frequency frequency hopping [4]	$n(1-\frac{1}{q})^{n-1}$	0	$q\, e^{-1}$
mobile users with pairwise transmission [4]	1	1	1
capture - power discrimination [4]	$1 \quad n=1$ $\frac{1}{\beta^2} \quad n>1$	$\frac{1}{\beta^2}$	$\frac{1}{\beta^2} + (1-\frac{1}{\beta^2})\exp(-\frac{\beta^2}{\beta^2-1})$
capture - timing discrimination [2]	$1 \quad n=1$ $(1-Q)^n \quad n>1$	0	$\max_{A>0} \{ (AQ-1)\, e^{-A} + e^{-AQ} \}$

Table 1. Examples.

3. Optimal control for the multipacket channel with bounded simultaneous successes

In this section we assume that there exists an upper bound on the number of packets that can be demodulated simultaneously and successfully, i.e. $\epsilon_{ij}=0$ if $j>K$. Note that in the case of a multipoint-to-point channel, such a limitation on the receiver capabilities is likely to happen in practice. For instance, systems with capture (see [5], [10]) are included in the case $K=1$. We assume here that the feedback is binary. More precisely, if slot n is empty then $Z_n=0$ and $Z_n=\bar{0}$ otherwise. As suggested by the results of Section 2, we consider retransmission probabilities of the form $p_n = A/S_n$. To fully take advantadge of the feedback information, it seems quite natural to consider a linear recursion for the backlog estimate, that is $S_{t+1} = \max\{\, A\,,\, S_t + \alpha\, I(Z_t=0) + \beta\, I(Z_t=\bar{0})\,\}$, where $I(E)$ denotes the indicator function of the event E.

Theorem 2. Assume that the new packet arrivals $(A_k)_{k\geq 0}$ are exponential type [6] (e.g. Poisson) and that the sequence $(C_n)_{n\geq 1}$ is nonincreasing and nonconstant. Denote by A the solution of (4) and $\eta_c = e^{-A}\sum_{n=1}^{\infty} C_n \frac{A^n}{n!}$. If $\alpha<0$ and $\beta>0$ verify the following two conditions,

C1. $\beta > \lambda$

C2. $\beta(1-e^{-A}) + \eta_c - \lambda + \alpha e^{-A} = 0$

then the control algorithm $\begin{cases} p_t = \dfrac{A}{S_t} \\ S_{t+1} = \max\{\, A\,,\, S_t + \alpha I(Z_t=0) + \beta I(Z_t=\bar{0})\,\} \end{cases}$ has a maximum stable throughput η_c.

Proof. The proof is based on the method developed in [14]. The idea is to use the properties of (X_n, S_n) to build a Lyapunov function whose drift is negative in the entire plane. Clearly, $(X_n, S_n)_{n\geq 0}$ is a homogeneous two dimensional vector Markov chain. Denote by $c(n,s)$ the backlog drift, $c(n,s) = E[X_{t+1} - X_t \mid X_t=n,\, S_t=s]$, by $\tilde{X}_t = S_t - X_t$ the error in the backlog estimate, and by $d(n,s) = E[\tilde{X}_{t+1} - \tilde{X}_t \mid X_t=n,\, S_t=s]$ its drift. Equation (2) being still valid here, we get for all $s \geq A$

$$c(0,s) = \lambda \tag{10}$$

$$c(n,s) = \lambda - \sum_{j=1}^{n} \binom{n}{j} \left(\frac{A}{s}\right)^j \left(1-\frac{A}{s}\right)^{n-j} C_j \qquad (n\geq 1)$$

Let us now compute the drift of the backlog estimate:

$$E[S_{t+1} - S_t \mid X_t=n,\, S_t=s] = \beta + (\alpha-\beta)\left(1-\frac{A}{s}\right)^n \qquad (s \geq A-\alpha) \tag{11}$$

$$E[S_{t+1} - S_t \mid X_t=n,\, S_t=s] = \beta + (A-s-\beta)\left(1-\frac{A}{s}\right)^n \qquad (A \leq s \leq A-\alpha)$$

It then follows from (10)-(11) that if $s\geq A-\alpha$:

$$d(0,s) = \alpha - \lambda \tag{12}$$

$$d(n,s) = \beta - \lambda + (\alpha-\beta)\left(1-\frac{A}{s}\right)^n + \sum_{j=1}^{n} \binom{n}{j} \left(\frac{A}{s}\right)^j \left(1-\frac{A}{s}\right)^{n-j} C_j \qquad (n\geq 1)$$

and if $A\leq s\leq A-\alpha$:

$$d(0,s) = A-s-\lambda \tag{13}$$

$$d(n,s) = \beta - \lambda + (A-s-\beta)\left(1-\frac{A}{s}\right)^n + \sum_{j=1}^{n} \binom{n}{j} \left(\frac{A}{s}\right)^j \left(1-\frac{A}{s}\right)^{n-j} C_j \qquad (n\geq 1)$$

Most of the properties of the drifts that are needed to build the Lyapunov function depend on the ratio $x=n/s$. For instance, if x is close to 1, the backlog estimate is close to its ideal value, and we should have $c(n,s) < 0$. Also, a well-behaved estimate should be such that $d(n,s) < 0$ (resp. $d(n,s) > 0$) for $x<1$ (resp. $x>1$). For this reason, an approximation of $c(n,s)$ and $d(n,s)$ in terms of x is needed. If ν is the difference $\eta_c - \lambda = \nu>0$, we can fix L such that $\sum_{L+1}^{\infty} A^n \frac{C_n}{n!} < \frac{\nu}{2}$, and so

$$\lambda < e^{-A} \sum_{n=1}^{L} C_n \frac{A^n}{n!} - \frac{\nu}{2} \tag{14}$$

Lemma 2. There exists two positive functions, $\epsilon(n,s)$ that goes to zero when either n or s goes to infinity, and $\epsilon'(n)$ that goes to zero when n goes to infinity, such that the following statements hold if $s>A$

$$(i)\ c(n,s) \leq F(x) + L\epsilon(n,s) \quad (n>L) \ \text{with} \ F(x) = \lambda - e^{-Ax} \sum_{j=1}^{L} (Ax)^j \frac{C_j}{j!}$$

$$(ii)\ d(n,s) \leq N(x) + \epsilon(n,s) + \epsilon'(n) \quad (n\geq 1,\ s\geq A-\alpha) \ \text{with} \ N(x) = \beta+\eta_c-\lambda + (\alpha-\beta)e^{-Ax}$$

$$(iii)\ d(n,s) \geq P_N(x) - N\epsilon(n,s) \quad (n>N) \ \text{with} \ P_N(x) = \beta-\lambda + (\alpha-\beta)e^{-Ax} + e^{-Ax} \sum_{j=1}^{N} (Ax)^j \frac{C_j}{j!}$$

Proof of Lemma 2. The proof follows from (10)-(13), Theorem 1 and Lemma 3 below, whose proof is omitted because of space limitations.

Lemma 3. Given an integer $L\geq 1$, we have for $n > L$, for $s > A$ and for all $j \in \{0, 1,..., L\}$:

$$(\frac{An}{s})^j \left| \frac{n(n-1)\cdots(n-j+1)}{n^j} (1 - \frac{A}{s})^{n-j} - e^{-\frac{An}{s}} \right| < \epsilon(n,s)$$

where $\epsilon(n,s)$ is a positive function going to zero when either n or s goes to infinity.

Define the following two regions in the (n,s) plane:

$$C(\lambda_0,\lambda_1) = \{ (n,s) \colon n\geq 0,\ s\geq 0,\ 1+\lambda_0 \leq \frac{n}{s} \leq 1+\lambda_1 \}$$

$$U_M = \{ (n,s) \colon n \geq M \text{ or } s\geq M \}$$

where λ_0 and λ_1 are any two real numbers such that $-\infty \leq \lambda_0 \leq \lambda_1 \leq +\infty$. We first show some simple geometric properties of these regions.

Lemma 4. Consider $\gamma>0$, $B>0$, and $\gamma-1<\lambda_0<\lambda_1<+\infty$; and assume that $|n-n'| \leq B$, $|s-s'| \leq B$, and $Q \geq \frac{B}{\gamma}(1+|\lambda_1|)(\lambda_1+2+\gamma)$. Then

1) $(n,s) \in C(\lambda_0,\infty) \cap U_Q \ => (n',s') \in C(\lambda_0-\gamma,\infty) \cap U_{Q-B}$

2) $(n,s) \in C(-\infty,\lambda_1) \cap U_Q \ => (n',s') \in C(-\infty,\lambda_1+\gamma) \cap U_{Q-B}$

3) $(n,s) \in C(\lambda_0,\lambda_1) \cap U_Q \ => (n',s') \in C(\lambda_0-\gamma,\lambda_1+\gamma) \cap U_{Q-B}$

Proof of Lemma 4. We show that the first statement holds, the proof of the other two being similar. Since $Q>B$, either $n\geq Q>B$, or $s\geq Q$, in which case $n > (1+\lambda_0) \frac{B}{\gamma} (1+|\lambda_1|)(2+\lambda_1+\gamma)$ which is strictly larger than B. Therefore $n>B$. Then

$$\frac{n'}{s'} \geq \frac{(n-B)(1+\lambda_0)}{n+B(1+\lambda_0)} = 1 + \frac{\lambda_0 n - 2B(1+\lambda_0)}{n+B(1+\lambda_0)}$$

so that $n'/s' \geq 1+\lambda_0-\gamma$ iff

$$\gamma n \geq B(1+\lambda_0)(2+\lambda_0-\gamma) \tag{15}$$

If $n\geq Q$, it is easily seen that (15) holds, and if $s\geq Q$ we have $n \geq (1+\lambda_0)Q > (1+\lambda_0) \frac{B}{\gamma} (1+\lambda_1)(2+\lambda_1+\gamma)$, from which (15) follows.

Let us now use the approximations of Lemma 2 to study some properties of the drifts in the plane.

Lemma 5. There exists constants $M > 0$, $\delta > 0$ and $\gamma \in]0,1/5[$ such that

(i) for all $(n,s) \in C(-5\gamma,5\gamma) \cap U_M$, $c(n,s) \leq -\delta$

(ii) for all $(n,s) \in C(-\infty,-\gamma) \cap U_M$, $d(n,s) \leq -\delta$

(iii) for all $(n,s) \in C(\gamma,+\infty) \cap U_M$, $d(n,s) \geq +\delta$.

Proof of Lemma 5.

(i) $F(x)$ is a continuous function and from (14) $F(1)<0$, so we can fix $\gamma \in]0,1/5[$ and $\delta_1 > 0$ such that $F(x) \leq -2\delta_1$ for all $x \in [1-5\gamma,1+5\gamma]$. Now if we choose $M>\max\{\frac{L}{1-5\gamma}, A(1+5\gamma)\}$ large enough so that $\epsilon(n,s) \leq \delta_1$ for $(n,s) \in U_M$ then we get from Lemma 2 (i): $c(n,s) \leq -\delta_1$ for all $(n,s) \in C(-5\gamma,5\gamma) \cap U_M$.

(ii) In $C(-\infty,-\gamma)\cap U_M$, $s\geq M$. Take $M\geq A-\alpha$. If $n=0$, $d(n,s)=\alpha-\lambda<0$. Fix N such that $\epsilon'(n)<|\frac{N(1-\gamma)}{3}|$ for $n>N$. For $1\leq n\leq N$, $d(n,s)\leq\beta-\lambda+(\alpha-\beta)(1-\frac{A}{s})^N+\frac{A}{s}N!\sum_{j=1}^{N}\frac{C_j}{j!}$, which goes to $\alpha-\lambda$ as s goes to infinity. Therefore we can take M large enough so that for $s\geq M$ and $1\leq n\leq N$, $d(n,s)\leq\frac{\alpha-\lambda}{2}$. $N(x)$ is a strictly increasing function and from C2 $N(1)=0$. So if $n>N$, we get from Lemma 2 (*ii*): $d(n,s)\leq\frac{2}{3}N(1-\gamma)+\epsilon(n,s)$ for $x<1-\gamma$.

(iii) If $s=A$ and $n\geq 1$, then $d(n,s)=\beta-\lambda+C_n>0$ from C1. If $s>A$ and $n>N$, Lemma 2 (*iii*) holds. Considering that

$$\frac{dP_N}{dx}/(Ae^{-A})=\beta-\alpha+C_1-(Ax)^N\frac{C_N}{N!}-\sum_{j=1}^{N-1}(Ax)^j\frac{C_j-C_{j+1}}{j!}$$

it can be seen that $\frac{dP_N}{dx}$ decreases from a positive value, $\beta-\alpha+C_1$, to $-\infty$. Therefore, $P_N(x)$ increases from $\alpha-\lambda<0$ to a positive maximum and then decreases towards $\beta-\lambda>0$. Denote by r_N the only zero of P_N. Clearly, $P_{N+1}\geq P_N$, so that (r_N) is a decreasing sequence; let r be its limit. It is easily shown that $\lim_{N\to\infty}P_N(r_N)=\beta-\lambda+(\alpha-\beta)e^{-Ar}+e^{-Ar}\sum_{j=1}^{\infty}(Ar)^j\frac{C_j}{j!}=0$, and that the equation $\beta-\lambda+(\alpha-\beta)e^{-Ax}+e^{-Ax}\sum_{j=1}^{\infty}(Ax)^j\frac{C_j}{j!}=0$ has only one solution. It then follows from C2 that $r=1$. Now fix N such that $r_N<1+\frac{\gamma}{2}$. Then $P_N(1+\gamma)>0$, so M can be chosen large enough so that (iii) holds.

From now on, M and γ are fixed so that Lemma 5 holds. Let us define the following Lyapunov function

$$V(n,s)=\max\{\,n,\ \frac{1+3\gamma}{3\gamma}(n-s),\ \frac{1-3\gamma}{3\gamma}(s-n)\,\}$$

$V(n,s)$ is equal to the first, second and third term inside the bracket when (n,s) is in $C(-3\gamma,3\gamma)$, $C(3\gamma,+\infty)$, and $C(-\infty,-3\gamma)$ respectively. We would like to have $E[V(X_{t+1},S_{t+1})-V(X_t,S_t)\mid X_t=n,S_t=s]\leq-\Delta$ for all $(n,s)\in U_M$, but this may fail if (X_{t+1},S_{t+1}) is not in the same region than (X_t,S_t).

Lemma 6. Given $\epsilon>0$, there exists some $\Delta>0$ and J_0 such that for $J>J_0$

$$E[V(X_{t+J},S_{t+J})-V(X_t,S_t)\mid X_t=n,S_t=s]\leq-\Delta J+\epsilon$$

for all $(n,s)\in U_{HJ}$, where H is a positive constant independent of J.

Proof of Lemma 6. Given any integer $J>0$, and any real number $\phi>0$, let us define the stopping time $\tau_{J,\phi}=\min\{s\geq 0\colon\sum_{k=t}^{s+t}A_k\geq\phi J\}$. ϕ can be chosen such that the event $\{\tau_{J,\phi}\leq J\}$ is unlikely.

Lemma 7. There exists $\phi>0$ such that $\lim_{J\to\infty}J\,P[\tau_{J,\phi}\leq J]=0$.

Proof of Lemma 7. Since $(A_k)_{k\geq 0}$ are exponential type, their common characteristic function $\psi(t)=E[e^{tA_k}]$ exists on an interval $[0,d]$. Denoting by $S_J=\sum_{k=1}^{J}A_{t+k}$, we get $\sum_{k=0}^{\infty}P[S_J=k]e^{tk}=\psi(t)^J$. Therefore $P[S_J\geq N]\leq\psi(t)^Je^{-Nt}$, so that if $N>J\frac{\log\psi(t)}{t}$ for some $t\in[0,d]$, we have $\lim_{J\to\infty}JP[S_J\geq N]=0$.

From now on, ϕ is fixed such that Lemma 7 holds, and we just write τ_J. Consider the decomposition

$$E[V(X_{t+J},S_{t+J})-V(X_t,S_t)\mid X_t=n,\ S_t=s]=$$

$$\sum_{k=0}^{J-1}E\{\,E[V(X_{t+k+1},S_{t+k+1})-V(X_{t+k},S_{t+k})\mid X_{t+k},\ S_{t+k}]I(\tau_J\leq J)\mid X_t=n,\ S_t=s\}\tag{16}$$

$$+\sum_{k=0}^{J-1}E\{\,E[V(X_{t+k+1},S_{t+k+1})-V(X_{t+k},S_{t+k})\mid X_{t+k},\ S_{t+k}]I(\tau_J>J)\mid X_t=n,\ S_t=s\}$$

Denote by T_1 and T_2 the two sums on the right hand side of (16). Since $|X_{t+1}-X_t|\leq K+A_t$ and $|S_{t+1}-S_t|\leq|\alpha|+\beta$, we have

$$|V(X_{t+1},S_{t+1})-V(X_t,S_t)|\leq\max\{\,1,\ \frac{1+3\gamma}{3\gamma},\ \frac{1-3\gamma}{3\gamma}\,\}(K+|\alpha|+\beta+A_t)\leq R(1+A_t)$$

where R is some constant independent of J. Therefore

$$T_1 \leq R(1+\lambda)JP[\tau_J \leq J] \tag{17}$$

so that from Lemma 7, T_1 goes to zero when J goes to infinity.

Lemma 8. If $\tau_J > J$, there exists $\mu>0$ independent of J such that for $0 \leq k \leq J-1$ and $(n,s) \in U_{Q(J)}$ $E[V(X_{t+k+1},S_{t+k+1}) - V(X_{t+k},S_{t+k}) \mid X_{t+k}, S_{t+k}] \leq -\mu$, where $Q(J)$ is a positive constant depending on J.

Proof of Lemma 8. To simplify notations, let us define $\Delta V_k = E[V(X_{t+k+1},S_{t+k+1}) - V(X_{t+k},S_{t+k}) \mid X_{t+k}, S_{t+k}]$, for $0 \leq k \leq J-1$. Since $\tau_J > J$, we have $\sum_{k=0}^{J-1} |X_{t+k+1} - X_{t+k}| \leq J(\phi+K)$ and $\sum_{k=0}^{J-1} |S_{t+k+1} - S_{t+k}| \leq J(|\alpha|+\beta)$. Set $B = J\max\{\phi+K, |\alpha|+\beta\}$, and define by $Q(J)$ any constant such that $Q \geq \max\{B+M, \frac{B}{\gamma}(1+4\gamma)(2+5\gamma)\}$.

Case 1: $(X_t,S_t) \in C(-2\gamma,2\gamma) \cap U_{Q(J)}$

From Lemma 4, $(X_{t+k},S_{t+k}) \in C(-3\gamma,3\gamma) \cap U_M \subseteq C(-5\gamma,5\gamma) \cap U_M$ for $0 \leq k \leq J-1$. Thus from Lemma 5, $\Delta V_k = c(X_{t+k},S_{t+k}) \leq -\delta$.

Case 2: $(X_t,S_t) \in C(4\gamma,\infty) \cap U_{Q(J)}$

Then from Lemma 4, $(X_{t+k},S_{t+k}) \in C(3\gamma,\infty) \cap U_M \subseteq C(\gamma,\infty) \cap U_M$, so from Lemma 5 $\Delta V_k = -\frac{1+3\gamma}{3\gamma}\, d(X_{t+k},S_{t+k}) \leq -\delta$.

Case 3: $(X_t,S_t) \in C(-\infty,-4\gamma) \cap U_{Q(J)}$

Since from Lemma 4, $(X_{t+k},S_{t+k}) \in C(-\infty,-3\gamma) \cap U_M \subseteq C(-\infty,-\gamma) \cap U_M$, we get $\Delta V_k = \frac{1-3\gamma}{3\gamma}\, d(X_{t+k},S_{t+k}) \leq -\frac{1-3\gamma}{3\gamma}\delta$.

Case 4: $(X_t,S_t) \in C(2\gamma,4\gamma) \cap U_{Q(J)}$

Then from Lemma 4, we have $(X_{t+k},S_{t+k}) \in C(\gamma,5\gamma) \cap U_M$, so from the definition of V, we get for $0 \leq k \leq J-1$:

$$V(X_{t+k+1},S_{t+k+1}) = \max\{X_{t+k+1}, \frac{1+3\gamma}{3\gamma}(X_{t+k+1}-S_{t+k+1})\}$$

$$V(X_{t+k},S_{t+k}) = \max\{X_{t+k}, \frac{1+3\gamma}{3\gamma}(X_{t+k}-S_{t+k})\}$$

so that another 4 cases have to be distinguished depending on the value of the two maxima above. Since both (X_{t+k+1},S_{t+k+1}) and (X_{t+k},S_{t+k}) belong to $C(-5\gamma,5\gamma) \cap C(\gamma,\infty) \cap U_M$, we have in any case from Lemma 5 $\Delta V_k \leq \max\{-\delta, -\frac{1+3\gamma}{3\gamma}\delta\} \leq -\delta$. For instance, if $X_{t+k+1} > \frac{1+3\gamma}{3\gamma}(X_{t+k+1}-S_{t+k+1})$ and $X_{t+k} < \frac{1+3\gamma}{3\gamma}(X_{t+k}-S_{t+k})$, we get

$$\Delta V_k = E[X_{t+k+1} - \frac{1+3\gamma}{3\gamma}(X_{t+k}-S_{t+k}) \mid X_{t+k},S_{t+k}] \leq c(X_{t+k},S_{t+k}) \leq -\delta$$

Case 5: $(X_t,S_t) \in C(-4\gamma,-2\gamma) \cap U_{Q(J)}$ is analogous to case 4.

We can now conclude the proof of Lemma 6. Define $H = \frac{(1+4\gamma)(2+5\gamma)}{\gamma}\max\{\phi+K, |\alpha|+\beta\}$. Choose J large enough so that $Q(J) = HJ$ and $P[\tau_J > J] > 1/2$. Then from (16) and Lemma 8, we get for $(n,s) \in U_{HJ}$: $T_2 \leq -\mu JP[\tau_J > J] < -\mu\frac{J}{2}$, which together with Lemma 7 and (17), ends the proof of Lemma 6.

Define the following

$$W_t = V(X_{tJ},S_{tJ})$$

$$F_t = \sigma\{A_{s-1}, X_s, 0 \leq s \leq t\}$$

$$a = HJ\max\{1, \frac{1+3\gamma}{3\gamma}, \frac{1-3\gamma}{3\gamma}\}$$

Lemma 9 (Hajek [6]). Let $\{W_t\}$ be a sequence of random variables adapted to an increasing family of σ-fields $\{F_t\}$. Suppose that W_0 is deterministic, that $\{W_t,F_t\}$ is exponential type and that for some $\epsilon > 0$ and $a > 0$ we have $E[(W_{t+1} - W_t + \epsilon)\, I(W_t > a) \mid F_t] \leq 0$ for all $t \geq 0$. Then for each value of W_0 the stopping time $\tau = \min\{t \geq 0; W_t \leq a\}$ is exponential type.

If $V(X_t,S_t) > a$, then $(X_t,S_t) \in U_{HJ}$, so we can apply Lemma 9 to our system to conclude that $\tau = \min\{t \geq 0, V(X_{Jt},S_{Jt}) \leq a\}$ is exponential type for any initial state. Since $V(X_t,S_t) \leq a$ implies that $X_t \leq a$ and $S_t \leq a/(1-\gamma)$, it follows that $\tau' = \min\{t \geq 0, X_t \leq a$ and $S_t \leq a/(1-\gamma)\}$ is also exponential type for any initial state, and from [9] that (X_t,S_t) is geometrically ergodic.

Note that Theorem 2 can be applied to the usual collision channel. To our knowledge, this is the only decentralized control algorithm yielding the optimal throughput e^{-1} with only a binary feedback. It is conjectured that Theorem 2 can be extended to the general reception matrix case. The drifts remain the same and Lemma 5 is still verified, but the proof of Lemma 6 seems rather difficult to extend since the number of successes per slot is no longer bounded.

Appendix A: Kaplan's condition

Consider a Markov chain with denumerable state space D, and one-step transition probability matrix $(P_{xy})_{(x,y)\in D}$. Let $V(x)$ be a Lyapunov function on D. Then the generalized Kaplan's condition holds if there exists a positive constant B such that for all $z \in [0,1[$ and all $x \in D$

$$z^{V(x)} - \sum_{y\in D} P_{xy} z^{V(y)} \geq -B(1-z)$$

1) One dimensional Kaplan's condition. Consider the model of Section 2 with a control scheme of the form $p_n = F(X_n)$, and the Lyapunov function $V(x)=x$. To check the Kaplan's condition, it is enough from [12] to show that the downward part of the drift, $D(i)$, is bounded below. For $i\geq 1$ and $1\leq k\leq i$ we have

$$P_{i,i-k} = \sum_{n=0}^{i-k} \lambda_n \sum_{j=k+n}^{i} \binom{i}{j} F(i)^j \, (1-F(i))^{i-j} \, \epsilon_{j,k+n}$$

after a change of variable, it follows that

$$D(i) = -\sum_{j=1}^{i} \binom{i}{j} F(i)^j \, (1-F(i))^{i-j} \sum_{n=0}^{j-1} \lambda_n \sum_{k=n+1}^{j} (k-n)\, \epsilon_{j,k} \tag{A-1}$$

a) If $p_n=\dfrac{A}{X_n}$, then Kaplan's condition holds. (A-1) yields in this case for $i\geq\max\{1, A\}$:

$$D(i) = -\sum_{j=1}^{i} \frac{i(i-1)\cdots(i-j+1)}{i^j} \frac{A^j}{j!} (1-\frac{A}{i})^{i-j} \sum_{n=0}^{j-1} \lambda_n \sum_{k=n+1}^{j} (k-n)\, \epsilon_{jk} \geq -Ae^A$$

b) If $(C_n)_{n\geq 1}$ is bounded, then Kaplan's condition holds independently of the retransmission policy. Denoting by U an upper bound for (C_n), (A-1) becomes

$$D(i) \geq -\sum_{j=1}^{i} \binom{i}{j} F(i)^j \, (1-F(i))^{i-j} \sum_{n=0}^{j-1} \lambda_n \, C_j \geq -\sum_{j=1}^{i} \binom{i}{j} F(i)^j \, (1-F(i))^{i-j} \, C_j \geq -U$$

2) Two dimensional Kaplan's condition. Consider now the general multipacket Aloha channel with a control algorithm of the form (1). Then (X_n, S_n) is the Markov chain of interest, and the relevant Lyapunov function is $V(n,s) = n$. We prove again that provided that $(C_n)_{n\geq 1}$ is bounded, Kaplan's condition holds. From [12], it is enough to show that the downward part $T(x)$ of the generalized drift is bounded below, with $T(x) = \sum_{y/V(y)<V(x)} P_{xy}\, (V(y)-V(x))$. Given a state $x=(i,s)$, we have

$$T(x) = -\sum_{r=1}^{i} r \sum_{k} P[X_{n+1}=i-r,\ S_{n+1}=k \mid X_n=i,\ S_n=s] = -\sum_{r=1}^{i} r\ P[X_{n+1}=i-r \mid X_n=i,\ S_n=s]$$

which is, in the same way as before

$$T(x) = -\sum_{r=1}^{i} r \sum_{n=0}^{i-r} \lambda_n \sum_{j=r+n}^{i} \binom{i}{j} (F(s))^j \, (1-F(s))^{i-j} \, \epsilon_{j,r+n} = -\sum_{j=1}^{i} \binom{i}{j} F(s)^j \, (1-F(s))^{i-j} \sum_{n=0}^{j-1} \lambda_n \sum_{r=n+1}^{j} (r-n)\, \epsilon_{jr}$$

this expression is similar to (A-1), and the end of the proof is the same.

References

[1] D. Bertsekas and R. Gallager, *Data Networks,* Prentice Hall, 1987.

[2] D.H. Davis and S.A. Gronemeyer, "Performance of Slotted Aloha Random Access with Delay Capture and Randomized Time of Arrival", *IEEE Trans. Commun.*, vol. COM-28, pp. 703-710, 1980.

[3] G. Fayolle, E. Gelenbe and J. Labetoulle, "Stability and Optimal Control of the Packet Switching Broadcast Channel", *J. Assoc. Comput. Machinery,* vol. 24, no. 3, pp. 375-386, 1977.

[4] S. Ghez, S. Verdú and S.C. Schwartz, "Stability of Multipacket Aloha", *Proc. Twenty First Conf. on ISS, The Johns Hopkins University,* March 87.

[5] D.J. Goodman and A.A.M. Saleh, "Local Aloha Radio Communications Systems with Capture and Packet Buffers", preprint, 1986.

[6] B. Hajek, "Hitting-time and Occupation-time Bounds implied by Drift Analysis with Applications", *Adv. Appl. Prob.*, vol. 14, pp. 502-525, 1982.

[7] B. Hajek and T. Van Loon, "Decentralized Dynamic Control of a Multiaccess Broadcast Channel", *IEEE Trans. Automatic Control,* vol. AC-27, pp. 559-569, 1982.

[8] M. Kaplan, "A Sufficient Condition for Nonergodicity of a Markov Chain", *IEEE Trans. Inform. Theory* , vol. IT-25, pp. 470-471, 1979.

[9] J.G. Kemeny, J.L. Snell and A.W. Knapp, *Denumerable Markov Chains,* Springer-Verlag, New York, 1976.

[10] C. Namislo, "Analysis of Mobile Radio Slotted Aloha Networks", *IEEE J. Select. Areas in Comm.,* vol. SAC-2, pp. 583-588, 1984.

[11] A.G. Pakes, "Some Conditions for Ergodicity and Recurrence of Markov Chains", *Opns. Res.,* vol. 17, pp. 1058-1061, 1969.

[12] L.I. Sennott, "Tests for the Nonergodicity of Multidimensional Markov Chains", *Opns. Res.,* vol. 33, pp. 161-167, 1985.

[13] M.K. Simon, J.K. Omura, R.A. Scholtz, B.K. Levitt, *Spread-Spectrum Communications* , Computer Science Press, 1985.

[14] J.N. Tsitsiklis, "Analysis of a Multiaccess Control Scheme", MIT LIDS Rep. LIDS-P-1534.

[15] W. Szpankowski, "Some Sufficient Conditions for Non-ergodicity of Markov Chains", *J. Appl. Prob.,* 22, pp. 138-147, 1985.

[16] S. Verdú, "Minimum Probability of Error for Asynchronous Gaussian Multiple-Access Channels", *IEEE Trans. Inform. Theory* , vol. IT-32, pp. 85-96, 1986.

STABILITY CONDITIONS IN A MULTIQUEUE SYSTEM WITH APPLICATIONS TO THE ANALYSIS OF COMPUTER COMMUNICATION NETWORKS

WOJCIECH SZPANKOWSKI* and VERNON REGO*

Department of Computer Science, Purdue University
West Lafayette, In 47907

1. A MULTIQUEUE MODEL

We consider a distributed system which consists of M distributed users that require the use of a single scarce resource. In queueing terminology, we say that customers in M queues compete for access to a single server. Each queue is of infinite capacity. The arrival process to the i-th queue is Poisson, with parameter λ_i, $i = 1, 2, \dots, M$. Messages arriving at the i-th queue possess independent lengths that form an i.i.d. random sequence with distribution $H_i(t)$, with average message length denoted by h_i, $i = 1, 2, \dots, M$. Each queue is served according to FIFO discipline. The server works in a distributed fashion. While visiting the i-th queue, the server serves at most s_i messages, and then goes to another queue according to some *unspecified* algorithm. The server may *stop at* a queue not serving any customers at all, even if the queue is not empty. If a queue is empty when the server visits it, naturally no customers are served. To avoid confusion, we coin the term *successful visit* if the server serves at least one message while visiting nonempty queues.

The next two examples specify possible distributed algorithms for server behavior.

EXAMPLE 1. *Token-passing system* [1]

In this system, M queues (users) are handled by a single token (server), which visits the queues in a cyclic order. It is assumed that a *walking time*, W_j, is required to switch from queue j to $(j+1) \bmod M$. More specifically, when the server visits the i-th queue, it serves at most s_i customers, then walks in time W_i to the $(i+1)$-st queue, etc.

EXAMPLE 2. *Buffered ALOHA system* [2] [3]

There are M distributed users, each having an infinite buffer capable of storing fixed-length packets. The packets are transmitted through a broadcast channel. The channel is slotted, and a slot duration is equal to a packet transmission time. Each nonempty user transmits a packet, with a probability r_i, $i = 1, 2, \dots, M$, in a slot. If two or more users transmit simultaneously, then a collision occurs and the packets must be retransmitted in the future. When exactly one packet is transmitted in a slot, then a successful transmission takes place. Referring to our multiqueue model, we say that the server (channel) successfully visits a queue, if and only if a successful transmission occurs. Note that the server may visit (stop-at) a nonempty queue, and neither a transmission nor a successful transmission may occur.

2. MAIN RESULT

Our multiqueue model can be described by an M-dimensional process $\mathbf{N}^t = (N_1^t, N_2^t, \dots, N_M^t)$, where N_j^t represents the j-th queue length at time t. We assume that such a representation of the model exists. The question is under what conditions will the process $\mathbf{N}^t$ be stable. By this we require that a stationary distribution of $\mathbf{N}^t$ as $t \to \infty$ exists and the distribution be honest. In other words, $\mathbf{N}^t$ is stable if for $\mathbf{x} \in Z^n$, where Z is a set of nonnegative integers, the following holds

$$\lim_{x \to \infty} \lim_{t \to \infty} \Pr\{\mathbf{N}^t < \mathbf{x}\} = \lim_{x \to \infty} \Pr\{\mathbf{N} < \mathbf{x}\} = 1 \tag{1}$$

This research for both authors was supported by the NSF under grant NCR - 8702115

Otherwise, it is unstable (for details see [4] [5] [6]).

To derive stability conditions for the above defined system, we need some more notation. Let $\mathbf{C}_{j,n}$, $j = 1, 2, \ldots, M$, $n = 0, 1, \ldots,$ denote the intervisit time between the n-th and the n+1-st successful visit of a server to the j-th queue. We call $\mathbf{C}_{j,n}$ a *cycle time* as suggested by Example 1. Let us also define a conditional cycle time $C^*_{j,n}$ as the interval of time between the n-th and the $(n+1)$-st successful visit of a server, to the j-th queue *under the assumption that the j-th queue at time n has at least* s_j *messages*, i.e., $N^n_j \geq s_j$. We assume

A1. The random sequence $\{C^*_{j,n}\}$ is strictly a stationary sequence with average C^*_j, i.e., $C^*_j = E\{C_{j,n} \mid N^n_j \geq s_j\}$.

Now we are ready to state our main result.

Theorem. Under the above assumptions, the process $\mathbf{N}^t = (N^t_1, \ldots, N^t_M)$ is stable, if and only if

$$\lambda_j C^*_j < s_j \quad \text{for all} \quad j = 1, 2, \ldots, M \tag{3}$$

To prove the theorem we need two lemmas.

Lemma 1. If for all $j = 1, 2, \ldots, M$, the one dimensional process N^t_j is stable, then the M-dimensional process $\mathbf{N}^t = (N^t_1, N^t_2, \ldots, N^t_M)$ is stable.

Lemma 2. If for some j, say j^*, $N^t_{j^*}$ is unstable, then $\mathbf{N}^t = (N^t_1, \ldots, N^t_M)$ is also unstable.

●

Proof of Theorem. We first prove sufficiency. Let us focus on one queue, say the j-th queue, and consider stability for that queue. This queue can be interpreted as an $M \mid G \mid s_j$ queue with general dependent service times $\{C_{j,n}\}^{\infty}_{n=0}$. By Loynes result [4], we know that it is stable if and only if (3) holds for a given j. By Lemma 1, we see that (3) is sufficient for stability of $\mathbf{N}^t$.
Necessity. Assume that (3) does not hold for at least one j, say j^*. Then, by Loynes result, the queue $N^t_{j^*}$ is unstable and by Lemma 2 the process $\mathbf{N}^t$ is also unstable.

□

3. APPLICATIONS

We discuss only the token passing system (Example 1). To prove stability we need to estimate the mean cycle time for each user, under the condition that this particular user has at least s_j messages. Let W_0 denote the average of the total walking time. Then, under the condition that $N^t_j \geq S_i$, the number of departing customers is equal to $\lambda_j\, C^*_j$, where C^*_j is defined as the conditional cycle time. Hence $C_j = \{W_0 + s_j h_j\}/\{1 - \rho_0 + \rho_j\}$ where $\rho_j = \lambda_j h_j$ and $\rho_0 = \sum_{j=1}^{M} \rho_i$. Surprisingly, in this case, we have been able to find the exact mean conditional cycle time. Hence, the Theorem gives sufficient and necessary conditions for a token passing system. This result was first stated by Kuehn [1], but however, without proof.

The ALOHA system (Example 2) is even more interesting since exact stability conditions are not known to date. Using our approach we are able to prove the exact stability conditions for two and three users, and very tight bounds for more users. This will be reported in a forthcoming paper.

References

[1] Kuehn, P.J., Multiqueue systems with nonexhaustive cyclic service, *The Bell System Technical Journal,* 58, 1979, pp. 671–698.

[2] Szpankowski, W., Stability conditions for multi-dimensional queueing systems with computer applications, to appear in *Operations Research,* 1987.

[3] Tsybakov, B., Mikhailov, W., Ergodicity of slotted ALOHA systems, *Prob. Peredachii Infor., 15, 1979, pp. 73–87.*

[4] Loynes, R., The stability of a queue with non-independent inter-arrival and service times, *Camb. Phios,* 58, 1962, pp. 497–520.

THROUGHPUT ANALYSIS OF MAXEMCHUK'S INTEGRATED VOICE/DATA LOCAL AREA NETWORK PROTOCOL †

RANJIT KOHLI*, ALAN C. MINGO**, KENNETH S. VASTOLA
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ABSTRACT

In [1], Maxemchuk developed a protocol allowing voice to be added, in a time division multiple access (TDMA) manner, to a local area network using the CSMA/CD (carrier sense multiple access with collision detection) protocol. The performance of this protocol was presented in [1] using simulation.

In this paper we develop the throughput performance using analytic techniques. Extensive numerical results are presented, and the trade-off between voice and data users is explored.

In this protocol, a data packet and the first voice packet of a talkspurt have equal priority in accessing the channel via CSMA/CD. The subsequent voice packets have higher priority using TDMA but must wait for any other packet heard to be transmitting during their scheduled time slot. This is the movable TDM slot feature discussed in [1].

In [4], the movable slot feature is retained but the first voice packets are given higher priority than data packets. This variation on Maxemchuk's protocol is analyzed for both throughput and delay in [4], and comparisons are given therein.

More details on the work presented here may be found in [2,3].

---

*Now with COMSAT Labs., Clarksburg, MD 20871.

**Now with IBM, Poughkeepsie, NY 12602.

†Research supported, in part, by an IBM Faculty Development Award and the RPI CIM Program.

REFERENCES

1. N.F. Maxemchuk, "A Variation on CSMA/CD That Yields Movable TDM Slots in Integrated Voice/Data Local Networks," Bell System Technical Journal, vol. 61, No. 7, Sept. 1982, pp. 1527-1550.
2. A. Mingo, "A Throughput Analysis of an Integrated Voice/Data LAN Protocol," Masters Thesis, Rensselaer Polytechnic Institute, Troy, New York, May 1985.
3. R. Kohli, "Analysis of a Protocol Allowing Mixed Voice and Data Packets in a LAN Environment," Masters Thesis, Rensselaer Polytechnic Institute, Troy, New York, May 1986.
4. Z. Gao and K.S. Vastola, "Analysis of an Integrated Voice/Data Local Network Protocol," Manuscript in Preparation.

THE EFFECT OF EVENLY DISTRIBUTED STATIONS ON CARRIER SENSE MULTIPLE ACCESS PROTOCOL

JAMES M. VANDERWIELE, JON G. BREDESON
AT&T Technologies, Inc., Oklahoma City, OK; University of Oklahoma, Norman, OK

1. ABSTRACT

A model for the CSMA data communications protocol is developed that lets the probability of a collision between two arrivals be dependent upon the physical distance between the two stations as well as the time interval between the two arrivals. The system throughput and average message delay both show improvement with such an analysis.

2. INTRODUCTION

The Carrier Sense Multiple Access (CSMA) protocol is one of several data communications schemes in use in the computer industry. Extensive analysis has been done to predict the behavior of CSMA networks. In these analyses, a "worst-case" assumption is made that the vulnerable time for any message is equal to the total end-to-end propagation delay of the network. In practice, however, the vulnerable time is a function of the distance between the two (or more) contending stations. This time can be viewed as a random variable and a more accurate analysis can be made by making some assumptions about this random variable. In this paper, the performance of CSMA networks is compared for the worst-case view of the vulnerable period and an estimate for this time. An analysis of CSMA with variable contention periods for the 1-persistent mode of operation will be presented as the primary focus. A simulation model that both verifies the analytical results and allows for further study of performance capabilities will be outlined. The results of this simulation work will be shown and comparisons made between deterministic and variable contention periods.

3. THROUGHPUT ANALYSIS WITH DISTRIBUTED STATIONS

The throughput analysis of a CSMA system with evenly distributed stations will now be developed. This analysis is similar to that done by Kleinrock and Tobagi.[1] One of the assumptions in this previous work is that the propagation delay is identical for every transmitter/receiver pair and is equal to the total end-to-end propagation time.

3.1 Summary of the Throughput Analysis

We would like to determine the throughput capability of the network under different levels of loading. The overall approach is to identify "busy" and "idle" periods on the network and calculate the amount of time that the network is busy compared to the total available time. We must also take into consideration the probability of finding the network in a given state. For example, if the network is lightly loaded, the probability of finding the network in an idle state is much greater than when the load is heavy.

The analysis of a system with distributed stations requires that some assumptions be made about the distribution of the stations and the contribution to the total network load that each station makes. We will have to determine the probability that a new arrival will collide with a message that has just begun its transmission. This probability is dependent upon the distance between these two stations and is no longer strictly a function of the two arrival times.

A basic throughput equation will be developed that is based on the expected lengths of busy and idle periods, the probabilities of finding the network in each of these states, and probabilities associated with interfering arrivals. We will then look at the parameters of the throughput equation individually as the calculation of many of them is fairly involved and combine all of the results at the end of the analysis. The final equation can then be easily programmed on a computer to be evaluated for various loads.

3.2 Assumptions

There are several general assumptions that we make for this analysis,including:

1. The time required to detect the carrier due to a packet being transmitted is negligible.
2. All packets are of constant length and therefore require a constant amount of time to transmit.
3. There are no errors caused by the channel, that is, the channel is noiseless.
4. The delay in changing from transmit to receive mode or vice versa is zero.
5. The average retransmission delay in the event of a collision is large compared to the packet transmission time.

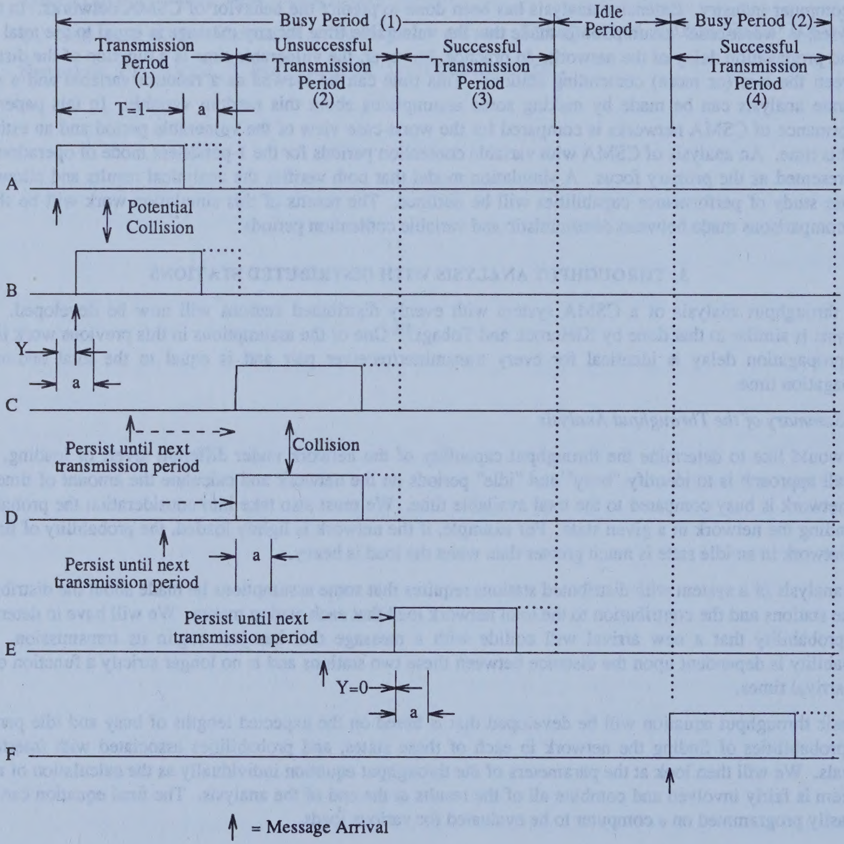
We shall let S represent the average rate of packet arrivals normalized to T, which is the length of time required to transmit one packet successfully. S therefore is expressed as the number of packets per transmission time. There will be some number of packets that are involved in collisions and are rescheduled. We will have a total offered rate of messages to the transmission medium that is greater than S and is given by G. G is the arrival rate of both new and retransmitted messages per packet transmission time. The steady state probability that any given arrival will be successfully transmitted, which will be denoted P_s, is given by

$$P_s = \frac{S}{G}.$$

The relationship between the total throughput of the system and the offered load is therefore given by

$$S = GP_s.$$

We shall let a represent the total propagation delay of the network normalized to T. a is the fraction of the packet transmission time equal to the end-to-end delay.



Busy Period (1)
Idle Period
Busy Period (2)
Transmission Period (1)
Unsuccessful Transmission Period (2)
Successful Transmission Period (3)
Successful Transmission Period (4)
T=1
a
A
Potential Collision
B
Y
a
C
Persist until next transmission period
Collision
D
a
Persist until next transmission period
Persist until next transmission period
E
Y=0
a
F
= Message Arrival

Figure 1. CSMA transmission times

We now make five additional assumptions that relate to the traffic source for the network and the distribution of stations along the transmission medium. The assumptions are:

1. There is one, continuous network. There are no subnetworks or branches involved.
2. The stations on the network will be spaced evenly along the entire length of the communications medium.
3. The number of stations, n, will be an even number. This is by no means a very constricting assumption

and simplifies some of the analysis.

4. The traffic source consists of an infinite number of users. These users collectively form a Poisson source. That is, the interarrival times of packets are independent and exponentially distributed and thus form a Poisson process.
5. While we are assuming that we have an infinite number of users, these users are located at n discrete stations (locations) with station 1 and station n located at opposite ends of the transmission line. There may be any number of users at a particular station.

Figures 1A through 1F show some of the definitions that will be used in formulating the analysis. In these figures, four transmission periods are shown. A transmission period is defined as the time from the onset of transmission of a message while no other message is active to the time when the original arrival, and any other arrivals that cause a collision with the first, have completed transmission. Any contiguous sequence of transmission periods is a busy period. There are two busy periods shown which are separated by an idle period. An idle period is the time between two busy periods.

In each of the six timing diagrams shown in Figure 1, there is one arrival to the system. The response of the system to that arrival depends upon the state of the system. Figure 1A represents an arrival to an idle system, so the message immediately begins transmitting. The time interval T, which is set equal to 1 for convenience, is the actual transmission time of the message. There is an additional time a that represents the delay until all stations know that a message has completed transmission. Figure 1A shows the start of both a transmission period and a busy period.

In Figure 1B, a second arrival has occurred during the first transmission period. This arrival is before the total propagation delay a. As is shown in the figure, there is a possible collision between the arrivals in 1A and 1B. For a CSMA model that specifies a constant contention interval, there would definitely be a collision. With variable contention, the collision depends upon the distance between the two stations. In the two figures, the messages are shown to collide and the transmission period ends after the arrival in 1B has completed transmission.

Figures 1C and 1D show two more arrivals during the first transmission period. These two do detect the carrier and thus are shown to persist until the end of the first transmission period. Since there is a message (or messages) persisting after transmission period one, a second transmission period begins with arrivals 1C and 1D colliding with probability 1. The second transmission period is therefore also unsuccessful.

The busy period continues due to the arrival in Figure 1E in the second transmission period. This arrival persists and begins transmission after arrivals 1C and 1D are finished and defines the start of the third transmission period. Since it was the only arrival in transmission period two and no messages arrive during transmission period three that could cause a collision, arrival 1E transmits successfully. After this transmission, no other messages are persisting so the busy period comes to an end. This defines the start of the idle period. The idle period continues until there is another arrival, as shown in Figure 1F. This last arrival successfully transmits during the fourth transmission period, which also is the entire second busy period.

There are a few variables that need to be defined. Let $\bar{B}$ represent the expected duration of a busy period. $\bar{I}$ is the expected duration of an idle period. $\bar{B}'$ represents the expected duration of a busy period excluding the time interval a of each transmission period.

The time between the first arrival of a transmission period and the last arrival that can cause a collision is given by the random variable Y. Y varies between 0 and a. A distribution function will be developed for Y based upon the probability of no arrivals in the time interval $a-y$.

q_0 is the probability that there are no arrivals during a transmission period that persist. In Figure 1, the first transmission period saw arrivals from 1C and 1D that persisted into transmission period two. Similarly, during transmission period two, 1E arrived and persisted into transmission period three. However, in transmission period three, there were no arrivals that could persist into a next transmission period. q_0 represents the probability of the scenario shown in transmission period three occurring.

Let $\hat{q}_0$ be the probability that no arrival occurs during a transmission period after only one arrival has occurred during that transmission period. This means that the message that arrived during the middle of a transmission period will be the only message persisting into the next transmission period. Referring to Figure 1, $\hat{q}_0$ will be the probability that after arrival 1E, there will be no other arrivals during transmission period two.

3.3 Basic Throughput Equation

The basic throughput equation is formulated by considering the three possible states that the network can be in when a packet arrives.

1. The idle state. In this case, no other station is attempting to transmit a packet. The arriving packet will be successful if no other packets arrive that can collide with it (which is a function of the time that another packet arrives and the distance that the two stations are apart).
2. The contention state. The maximum time for this interval is a. In this state, the packet arrives after another station has begun transmitting but before the packet's station has detected the carrier. There will be a collision and both packets will be rescheduled.
3. The transmission state. The packet arrives to the network and is able to detect the carrier of another packet that is already being transmitted. The packet will persist until the carrier is no longer sensed and then attempt transmission. The packet will be successfully transmitted if
    a. No other packet arrives during the current transmission period, and
    b. When the packet that arrived to the transmission state begins its own transmission period, no other packets arrive that collide.

The probability of successful transmission, P_s, is equal to the sum of the probability of finding the network in a given state multiplied by the probability of a successful transmission from that state, or

$$P_s = \sum_{i=1}^{3} P_i \, P_{success\,|\,i}$$

where i represents one of the three states of the system (idle, contention, or transmission).

First, consider the probabilities of finding the network in a given state. Since an arrival can find the network in only one of the three states, they may be considered mutually exclusive events. The probability of finding the network in the idle state is

$$\frac{\overline{I}}{\overline{I} + \overline{B}}$$

which is the expected idle time of a cycle divided by the total time of a cycle. The probability of finding the network in the transmission state is

$$\frac{\overline{B}'}{\overline{I} + \overline{B}}$$

and represents the busy time, excluding the contention period, divided by the total time. Finally, the probability of finding the network in the contention state is

$$\frac{\overline{B} - \overline{B}'}{\overline{I} + \overline{B}}$$

which represents only the potential contention interval for each transmission period.

Now the probabilities associated with the successful transmission of a packet from a particular state must be considered. The probability that a packet transmits successfully from the idle state will be denoted P_N, which is the probability of no collision. The probability that a packet transmits successfully from the transmission state is $\hat{q}_0$, which was defined previously, multiplied by P_N. $\hat{q}_0$ is multiplied by P_N to take into account the fact the there must be no interfering arrivals in the message's transmission period. Finally, the probability that a packet transmits successfully from the contention state will be simply zero. The basic throughput equation is thus

$$P_s = \frac{\overline{I}}{\overline{I} + \overline{B}} P_N + \frac{\overline{B}'}{\overline{I} + \overline{B}} \hat{q}_0 P_N = \left[\frac{\overline{I}}{\overline{I} + \overline{B}} + \frac{\overline{B}'}{\overline{I} + \overline{B}} \hat{q}_0 \right] P_N = M \, P_N \, .$$

Each of the values in this equation will be derived independently with the exception of $\overline{I}$. If the average arrival rate of messages is given by G, then $1/G$ is the average time between messages or the average idle time. $\overline{I}$ is given by

$$\overline{I} = \frac{1}{G} \, .$$

3.4 Calculation of The Parameters for the Basic Throughput Equation

3.4.1 Calculation for P_N. If we consider a CSMA system in which any arrival that occurs during the first a of a transmission period causes a collision, P_N is given by

$$P_N = \frac{e^{-Gt}\,(Gt)^n}{n\,!}\,\Big|_{n=0,t=a} = e^{-Ga}\,.$$

By considering the probability of an arrival from each station, a good estimate of P_N for variable contention can be derived and is given by

$$P_N = \frac{2}{n}\sum_{g=1}^{\frac{n}{2}}\left[\prod_{h=1}^{n-g} e^{-\frac{Ga(h)}{n(n-1)}}\prod_{f=0}^{g-1} e^{-\frac{Ga(f)}{n(n-1)}}\right].$$

This expression is the probability that for any arrival, there will be no other messages arrive that will cause a collision after the arrival begins its transmission. The equation is a bit bulky, but it can easily be evaluated by a computer and called as a function.

3.4.2 Calculation of $\bar{B}$ and $\bar{B}'$. $\bar{B}$ is the expected duration of a busy period where a busy period is defined as all of the activity occurring on the network between any two idle periods. The busy period consists of a series of transmission periods. The expected time of the busy period is the expected duration of one transmission period multiplied by the expected number of transmission periods in a busy period.

q_0 has been defined as the probability that there are no accumulated arrivals at the end of a transmission period. The number of transmission periods in a single busy period can be represented by a geometric distribution with q_0 equal to the probability of a success. Here, a success implies that there are no accumulated arrivals which means that the busy period has come to an end. The expected value of such a geometric distribution is one divided by the probability of success. The expected value of the number of transmission periods is thus one divided by the probability of no accumulations, which is equal to $1/q_0$.

Referring to Figure 1, a single transmission period consists of the transmission time $1+a$ and Y, which can vary from 0 to a. The expected value of a transmission period is thus $1+a+\bar{Y}$, where $\bar{Y}$ is the expected value of Y. $\bar{B}$ can be formulated as

$$\bar{B} = (1+a+\bar{Y})\frac{1}{q_0} = \frac{1+a+\bar{Y}}{q_0}\,.$$

Similarly, $\bar{B}'$ is given by

$$\bar{B}' = \frac{1+a+\bar{Y}-a}{q_0} = \frac{1+\bar{Y}}{q_0}\,.$$

Formulas for the expected duration of Y, or $\bar{Y}$, and the probability of no accumulated arrivals in a transmission period, q_0, must now be derived.

3.4.3 Calculation of $\bar{Y}$. The determination of an expression for the expected value of Y requires that we identify a distribution for Y. We must find the probability that no arrival occurs in time $a-y$ that will cause interference. The distribution function can be written as

$$\begin{aligned} F_Y(y) &= Pr(no\ arrival\ in\ a-y) + Pr(arrival\ in\ a-y\ but\ no\ interference) \\ &= 1 - Pr(arrival\ in\ a-y\ AND\ interference) \end{aligned}$$

since the probability of having interference and no arrival is zero. This equation can be written

$$F_Y(y) = 1 - \Big[Pr(interference\,|\,arrival\ in\ a-y)\,Pr(arrival\ in\ a-y)\Big]\,.$$

The second probability function in this equation is

$$Pr(arrival\ in\ a-y) = 1 - Pr(no\ arrival\ in\ a-y) = 1 - e^{-G(a-y)}\,.$$

For a CSMA system in which all delays between stations are a, the distribution becomes

$$F_Y(y) = 1 - (1)\Big[1 - e^{-G(a-y)}\Big] = e^{-G(a-y)}\,.$$

We can approximate the probability of interference given an arrival in $a-y$ as $(a-y)/a$. This represents an approximation of the number of stations that will not have detected the presence of a transmission at time y after the start of that transmission. The distribution for Y is thus given by

$$F_Y(y) = 1 - \left[\frac{a-y}{a}\right]\Big[1 - e^{-G(a-y)}\Big] = \frac{y}{a} + \frac{a-y}{a}\,e^{-G(a-y)}\,.$$

The expected value of Y is found from

$$\overline{Y} = \int_0^a y\, f_Y(y)\, dy$$

where $f_Y(y)$ is the first derivative of $F_Y(y)$. The integration is over the range 0 to a since y is only defined in that interval. After performing the integration we have

$$\overline{Y} = \frac{a}{2} + \frac{1}{G}\, e^{-Ga} - \frac{1}{G^2 a}\left[1 - e^{-Ga}\right]$$

which can be substituted into the previous equations for $\overline{B}$ and $\overline{B}'$.

3.4.4 Calculation of q_0. As stated previously, q_0 is the probability that there are no packets accumulated at the end of a transmission period. This can be interpreted as the probability there are no arrivals that will defer into the next transmission period. This can be expressed as

$$q_0 = Pr\,(no\ arrival\ that\ defers) = 1 - Pr\,(arrival\ that\ defers)\,.$$

Looking at the time intervals $1{+}Y$ and $a-Y$ individually, q_0 is given by

$$q_0 = 1 - \Big[\,Pr\,(defer \mid arrival\ in\ 1{+}Y)Pr\,(arrival\ in\ 1{+}Y) + Pr\,(defer \mid arrival\ in\ a{-}Y)Pr\,(arrival\ in\ a{-}Y)\Big]$$

which simplifies to

$$q_0 = Pr\,(no\ arrival\ in\ 1{+}Y) - Pr\,(defer \mid arrival\ in\ a{-}Y) \times (1 - Pr\,(no\ arrival\ in\ a{-}Y))$$

The first term of this equation is given by

$$Pr\,(no\ arrivals\ in\ 1{+}Y) = \int_0^\infty e^{-G(1+Y)}\, dF_Y(y) = \int_0^a e^{-G(1+Y)}\, dF_Y(y)$$
$$= \frac{e^{-G}}{Ga} + e^{-G(1+a)}\left[\frac{Ga}{2} - \frac{1}{Ga}\right]$$

The probability that an arrival in the time period $a-Y$ defers can be estimated by

$$Pr\,(defer \mid arrival\ in\ a{-}Y) = \int_0^a \frac{y}{a}\, dF_Y(y) = \frac{\overline{Y}}{a.}$$

The last of the three quantities is the probability of an arrival in $a-Y$. This can be expressed as

$$Pr\,(no\ arrival\ in\ a{-}Y) = e^{-Ga}\int_0^a e^{Gy}\, dF_Y(y) = \frac{e^{-2Ga}}{2} - \frac{e^{-Ga}}{Ga} + \frac{e^{-2Ga}}{4Ga} + \frac{3}{4Ga}\,.$$

Compiling the three results obtained above, the final equation for q_0 is

$$q_0 = \frac{e^{-G}}{Ga} + e^{-G(1+a)}\left[\frac{Ga}{2} - \frac{1}{Ga}\right] - \left[\frac{\overline{Y}}{a}\right]\left[1 - \left[\frac{e^{-2Ga}}{2} - \frac{e^{-Ga}}{Ga} + \frac{e^{-2Ga}}{4Ga} + \frac{3}{4Ga}\right]\right].$$

This equation can also be substituted into those given previously for $\overline{B}$ and $\overline{B}'$.

3.4.5 Calculation of $\hat{q}_0$. This is the last value that is needed. Recall that $\hat{q}_0$ is the probability that a message that arrives during a transmission period and after the contention interval is the only message arriving during that period. We need to define

$$Z = 1 + Y\,,\quad z = 1 + y\,,\quad y = z - 1$$

which will be used in the development. Also, let $\hat{Z}$ be the length of the transmission period (not including the first a) in which an arrival occurs.

If we examine the random variable $\hat{Z}$ closely, we see that it does not have the same distribution as Z. We stated earlier that a busy period consists of a sequence of transmission periods where each transmission period is equal to Z plus a contention interval a. $\hat{Z}$, however, represents the length of a specific Z in which an arrival has occurred. An arrival is more likely to occur in a longer segment Z than a shorter one. This likelihood of a specific arrival occurring in a longer segment skews the distribution of $\hat{Z}$. For a Poisson process, the relationship between the density function for $\hat{Z}$ and the density function for Z can be expressed by[2]

$$f_{\hat{Z}}(z) = \frac{z\, f_Z(z)}{\bar{Z}} .$$

By determining the density function for Z, finding the density for $\hat{Z}$ from that for Z, and then calculating $\hat{q}_0$ we have,

$$(1+\bar{Y})\,\hat{q}_0 = \int_1^{1+a} e^{-Gz}\, z\, f_Z(z)\,dz = e^{-G(1+a)}\left[\frac{1}{Ga} - \frac{1}{G} - \frac{1}{G^2 a} + \frac{Ga^2}{6} + \frac{Ga}{2} - \frac{a}{2}\right] + e^{-G}\left[\frac{1}{Ga} + \frac{1}{G^2 a}\right] .$$

This equation is expressed as the product of $\hat{q}_0$ and $1+\bar{Y}$ because that product shows up in the final equation so there is no benefit in evaluating $\hat{q}_0$ alone.

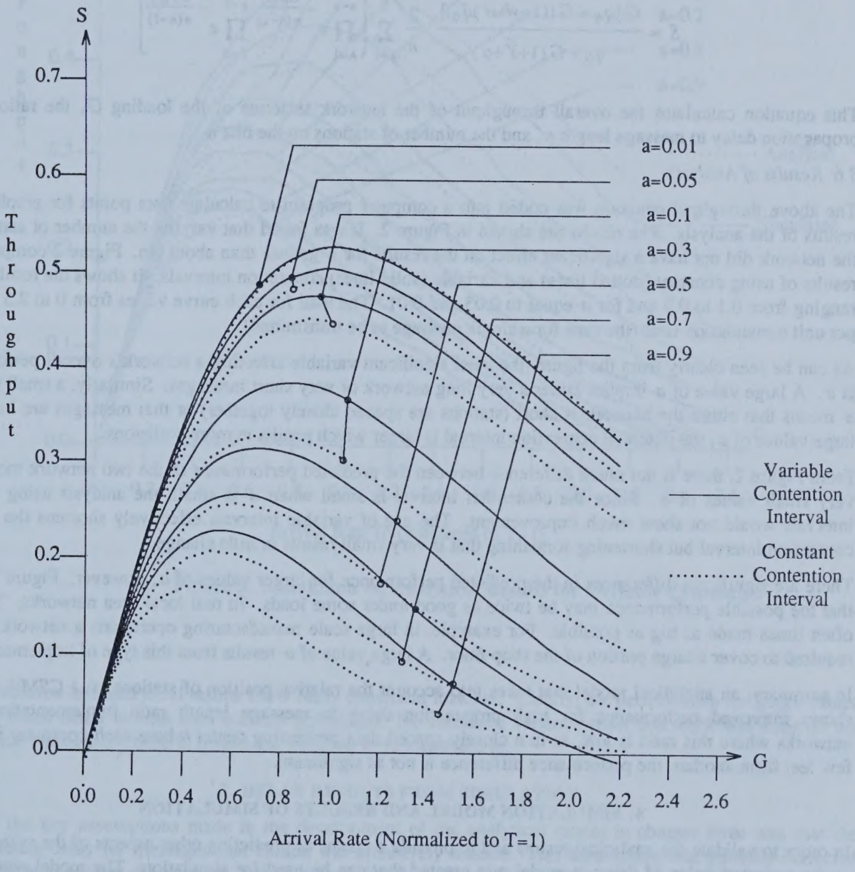


Figure 2. Comparison of Contention Intervals

3.5 Assembling the Basic Throughput Equation

The basic throughput equation can now be found by substituting the results for all of the parameters of that equation. Recall that

$$S = GP_S\ , \quad S = G\, P_N\, M$$

Solving for M,

$$M = \frac{\bar{I}}{\bar{B}+\bar{I}} + \frac{\bar{B}'\hat{q}_0}{\bar{B}+\bar{I}}$$

$$M = \frac{\frac{1}{G}}{\frac{1+\bar{Y}+a}{q_0}+\frac{1}{G}} + \frac{\frac{1+\bar{Y}}{q_0}\hat{q}_0}{\frac{1+\bar{Y}+a}{q_0}+\frac{1}{G}} = \frac{q_0 + G\,[(1+\bar{y})\hat{q}_0]}{q_0 + G\,(1+\bar{Y}+a)} .$$

The final result is thus given by

$$S = \frac{G\,[q_0 + G\,((1+ybar)\hat{q}_0)]}{q_0 + G\,(1+\bar{Y}+a)} \; \frac{2}{n} \sum_{g=1}^{\frac{n}{2}} \left[\prod_{h=1}^{n-g} e^{-\frac{Ga(h)}{n(n-1)}} \prod_{f=0}^{g-1} e^{-\frac{Ga(f)}{n(n-1)}} \right] .$$

This equation calculates the overall throughput of the network in terms of the loading G, the ratio of the propagation delay to message length a, and the number of stations on the line n.

3.6 Results of Analysis

The above throughput equation was coded into a computer program to calculate data points for graphing the results of the analysis. The results are shown in Figure 2. It was found that varying the number of stations on the network did not have a significant effect on the results for n greater than about ten. Figure 2 compares the results of using constant (dotted lines) and variable (solid lines) contention intervals. It shows the results for a ranging from 0.1 to 0.9 and for a equal to 0.05 and 0.01. The load for each curve varies from 0 to 2.5 arrivals per unit transmission time (the time for a single message to be transmitted).

As can be seen clearly from the figure, the most significant variable affecting a network's overall performance is a. A large value of a implies either a very long network or very short messages. Similarly, a small value of a means that either the network is short (stations are spaced closely together) or that messages are long. For large values of a, the potential contention interval is larger which results in more collisions.

From Figure 2, there is not much difference between the predicted performance of the two network models for very small values of a. Since the contention interval is small when a is small, the analysis using variable intervals would not show much improvement. The use of variable intervals effectively shortens the average contention interval but shortening something that is very small results in little change.

There are significant differences in the predicted performance for larger values of a, however. Figure 2 shows that the possible performance may be twice as good under some loads. In real local area networks, "local" is often times made as big as possible. For example, in large scale manufacturing operations a network may be required to cover a large portion of the shop floor. A large value of a results from this type of implementation.

In summary, an analytical model that takes into account the relative position of stations on a CSMA network shows improved performance for high propagation delay to message length ratio implementations. For networks where this ratio is low, as in a closely spaced data processing center where each computer is only a few feet from another, the performance difference is not as significant.

4. SIMULATION MODEL AND RESULTS OF SIMULATION

In order to validate the analytical results and to provide a means of predicting other aspects of the system, such as the expected value of delay, a model was created that can be used for simulation. The model was created using the same assumptions, such as those made for the network's operation and the message arrival distribution, that were used in the analytical portion to do comparisons between the analytical and simulated results.

The simulation model consists of a message generation block, a station assignment block that allows us to emulate many evenly distributed stations, a detect/collide checker that compares the arrival time of a message with those that are currently being transmitted, a check block to ensure that only one message has been transmitted, and a statistics block to collect information to calculate throughput and average delay.

Figure 3 compares the simulated results with those derived analytically. On the curve, the dashed lines represent analytic data and the solid lines represent simulation results. The curve is for a ranging from 0.1 to 0.9 and the load from 0 to 2.5 messages per transmission time. The number of stations is set at 30.

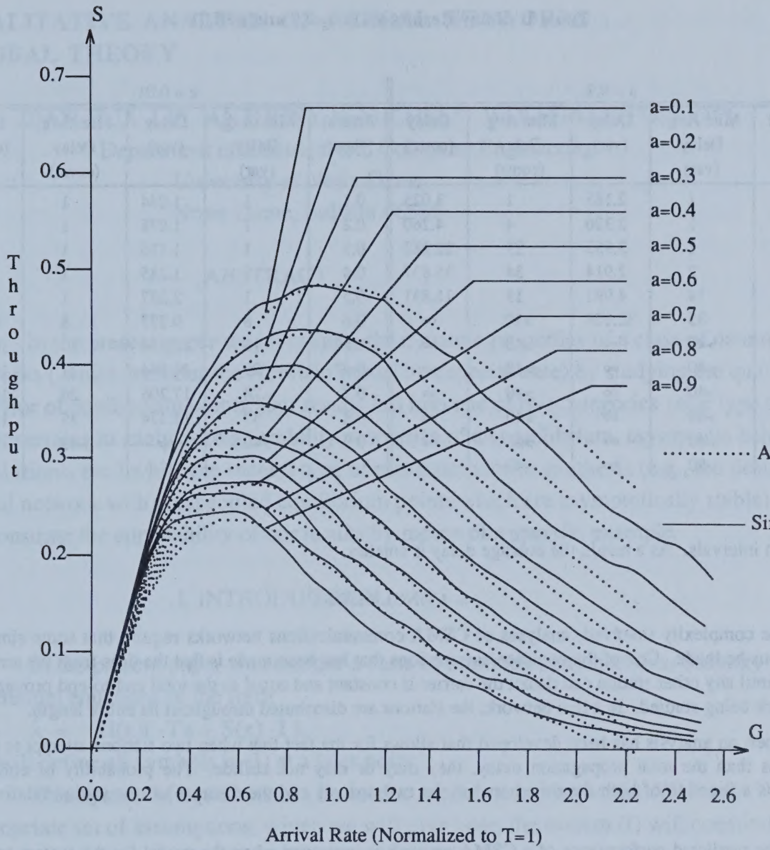


Figure 3. Simulation vs. Analytical Results for Variable Contention

The simulation and analytical results show fairly consistent matches. Clearly, the performance for larger values of a is much better in both with variable contention. No extensive variance reduction work was done on the simulation because the general results are what we were interested in.

5. DELAY RESULTS FROM SIMULATION

One of the key assumptions made in the development of the analytical model in chapter three was that the rescheduling delay for messages that collide was effectively infinite. This assumption was partially corrected by providing a load to the network that represented both new and rescheduled arrivals. The simulation model was written to incorporate this same assumption in order to compare the results of the two models. Since the simulation and analytical models yield consistent results, the simulation should be providing a good picture of the operation of the network. While modifying the delay assumption in the analytical model would invalidate it, the simulation model can be changed to incorporate a rescheduling delay to extend the investigation of variable contention intervals.

Table 1 shows the results of these simulations. In this table, the dependency of the delay on a is similar to that for the throughput. Also, the impact of allowing for variable contention intervals is smaller for a=0.01. In the delay results, there cannot really be any positive conclusions drawn from the data for very small values of a. However, for a=0.9, the minimum value for the mean of the rescheduling distribution is less with variable

Table 1. Delay Results With a=0.9 and a=0.01

a = 0.9					a = 0.01				
Arrival Rate	Min. Avg. Delay (var)	Delay (var)	Min. Avg. Delay (const)	Delay (const)	Arrival Rate	Min. Avg. Delay (var)	Delay (var)	Min. Avg. Delay (const)	Delay (const)
0.1	1	2.185	1	3.025	0.1	1	1.044	1	1.044
0.15	1	2.320	4	4.260	0.2	1	1.078	1	1.078
0.18	1	2.953	29	22.282	0.3	1	1.136	1	1.136
0.19	2	2.914	34	35.837	0.4	1	1.285	1	1.285
0.2	14	4.991	19	11.831	0.5	1	2.257	1	1.811
0.3	35	32.109	>40	∞	0.6	8	9.237	8	9.516
0.4	>40	∞	>40	∞	0.62	13	12.454	17	10.873
0.5	>40	∞	>40	∞	0.67	20	14.844	21	16.112
0.6	>40	∞	>40	∞	0.7	28	17.206	26	19.098
0.7	>40	∞	>40	∞	0.8	37	13.374	35	15.988
0.8	>40	∞	>40	∞	0.9	>40	∞	>40	∞
0.9	>40	∞	>40	∞					

contention intervals. As a result, the average delay is smaller.

6. CONCLUSIONS

Due to the complexity involved, analyses of CSMA communications networks require that some simplifying assumptions be made. One of the common assumptions that has been made is that the time from the arrival of a message until any other station can detect the carrier is constant and equal to the total end-to-end propagation of the network being studied. In a live network, the stations are distributed throughout its entire length.

In this paper, an analysis has been developed that allows for the fact that when two stations attempt to transmit within less than the total propagation delay, they may or may not collide. The probability of collision or detection is a function of both the time between the two arrivals and the distance of one station relative to the other.

Overall, the predicted performance of a CSMA network is improved when the model for the system takes into account the fact that the stations will be distributed along the transmission medium. When the network is constructed such that it is operating under near optimal conditions, the impact is not as great. In other circumstances, the results indicate significant improvement.

REFERENCES

1. Leonard Kleinrock and Fouad A. Tobagi, "Packet Switching in Radio Channels: Part I-Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Transactions on Communications*, volume Com-23, 1975, pp. 1400-1416.

2. Leonard Kleinrock, *Queueing Systems, Vol. I: Theory*, (New York: Wiley Interscience, 1975), pp. 169-174.

QUALITATIVE ANALYSIS OF NEURAL NETWORKS : GLOBAL THEORY

JIAN-HUA LI* ANTHONY N. MICHEL* WOLFGANG POROD**
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

In the present paper we investigate the dynamic properties of a class of neural networks (which includes the Hopfield model as a special case) by studying the qualitative behavior of equilibrium points. Our results fall into one of two categories : one type of result pertains to analysis (e.g., stability properties of an equilibrium, asymptotic behavior of solutions, etc.) while the second type of result pertains to synthesis (e.g., the design of a neural network with prespecified equilibrium points which are asymptotically stable). We demonstrate the applicability of our results by means of a specific example.

I. INTRODUCTION

In the present paper we consider a class of nonlinear, autonomous, ordinary differential equations of the form

$$\dot{x} = - H(x)(-Tx + S(x) -I). \tag{I}$$

We will define all symbols in (I) at a later point.

This system of equations can be used to model neural networks. With an appropriate set of assumptions, which we will give later, the system (I) will constitute a slight generalization of the Hopfield model [1].

The structure of this paper is as follows. In Section 2, we establish the notation used throughout this paper. In Section 3, we present the system which we study and we enumerate the assumptions made for this system. In Section 4, we state the results under the category Analysis while in Section 5, we establish the results under the category Synthesis. The applicability of the results of Section 5 is demonstrated by means of a specific example in Section 6. The paper is concluded with a few pertinent remarks in Section 7.

---

*The work of J.Li and A.N.Michel was supported by the National Science Foundation under grant ECS 84-19918.

**The work of W.Porod was supported by SDIO/IST and managed by the ONR under grant N00014-86-k 0506.

II. NOTATION

Let V and W be arbitrary sets. Then $V \cup W$, $V \cap W$, $V-W$ and $V \times W$ denote the union, intersection, difference and Cartesian product of V and W , respectively. If V is a subset of W , we write $W \supset V$ and if x is an element of V, we write $x \in V$. If f is a function from V into W , we write $f:V \rightarrow W$ and we let $f(U)=\{f(x) \in W : x \in U\}$ for $V \supset U$, and $f^{-1}(y)= \{x \in V : f(x)=y \}$ for $y \in W$. Let **R** denote the set of real numbers. If $V_1,\cdots,V_n$ are n arbitrary sets, their Cartesian product is denoted by $\prod^n{}_{i=1}V_i=V_1 \times \cdots \times V_n$. If in particular, $V=V_1=\cdots=V_n$, we write $\prod^n{}_{i=1}V_i=V^n$. Let $\mathbf{R}^n$ be real n-space. If $x \in \mathbf{R}^n$, then $x^T=(x_1,\cdots,x_n)$ denotes the transpose of x. When using a norm for $x \in \mathbf{R}^n$, $|x|$, we will have in mind $|x|=\max_{1 \leq i \leq n}\{|x_i|\}$. If $x \in \mathbf{R}^n$ and $Y \subset \mathbf{R}^n$, then $x \perp Y$ will mean that $x^T \cdot y=0$ for all $y \in Y$. If $A=[a_{ij}]$ is an arbitrary matrix, then A^T denotes the transpose of A . If A is a symmetric matrix, by $A>0$ we mean that A is positive definite and by $A \geq 0$ we mean that A is positive semi-definite. A function $F:V \rightarrow W$, where $\mathbf{R}^n \supset V, \mathbf{R}^m \supset W$, is said to be of class C^k or a C^k-function if for each component of F, the k-th order derivative exists and is continuous .

Given a C^2 -function $g:V \rightarrow \mathbf{R}$ where $\mathbf{R}^n \supset V$, we denote the gradient of g by $\nabla g(x)=(\partial g_1(x),\cdots,\partial g_n(x))^T$ and we denote the $n \times n$ Jacobian matrix of g by $J_g(x)= [\partial^2 g(x)/\partial x_i \partial x_j]$. An element $x \in V$ is called a critical point of g if $\nabla g(x)=0$. Also, $x \in V$ is said to be a local minimum of g if there is an open neighborhood U of x such that for all $y \in U$, $g(y) \geq g(x)$.

III. NEURAL NETWORK MODEL

We will consider neural networks described by differential equations of the form

$$\dot{x} = -H(x)(-Tx + S(x) - I) \qquad \text{(I)}$$

where $x=(x_1,\cdots,x_n)^T \in (-1,1)^n$, $\dot{x}=dx/dt$, H is a function from $(-1,1)^n$ into $\mathbf{R}^{n \times n}$ (i.e. for each $x \in (-1,1)^n$, H(x) is an $n \times n$ matrix), $T=[T_{ij}]$ is an $n \times n$ constant matrix, $S(x)=(s_1(x_1),\cdots,s_n(x_n))^T$ where $s_i : (-1,1) \rightarrow \mathbf{R}$ and $I=(I_1,\cdots,I_n)^T$ is a constant real vector. In subsequent assumptions, we will impose restrictions on the functions H(x) and S(x) and on the matrix T.

Remark 3.1. In [1], Hopfield considers a continuous-variable neural system which is realizable by electrical circuits and which is represented by the system of equations

$$C_i(du_i/dt) = \Sigma^n{}_{j=1}T_{ij}v_j - u_i/R_i + I_i \qquad ,i = 1, \cdots, n \qquad \text{(H)}$$

where for each i, C_i, R_i and I_i are constants, $C_i>0$ and $R_i>0$, $[T_{ij}]$ is an $n \times n$ symmetric constant matrix, $u_i \in \mathbf{R}$, $v_i \in (-1,1), v_i=g_i(\lambda u_i)$ where $\lambda>0$ is a constant. The function g_i is assumed to have the following properties: g_i: $\mathbf{R} \rightarrow (-1,1)$ is monotone-increasing, $g_i(\rho) = -g_i(-\rho)$, the inverse g_i^{-1}: $(-1,1) \rightarrow \mathbf{R}$ exists, and g_i and g_i^{-1} are C^1-functions. When we represent (H) in terms of the variables v_i, $i=1,\cdots,n$, we have

$$dv_i/dt = (dv_i/du_i)(du_i/dt)=$$
$$=-(\lambda/(dg_i^{-1}(v_i)/dv_i))(1/C_i)(-\Sigma^n{}_{j=1}T_{ij}v_j+g^{-1}(v_i)/(\lambda R_i)-I_i), i=1,\cdots,n. \qquad \text{(H')}$$

If in (H'), we let $(v_1,\cdots,v_n)^T=x$, $(g^{-1}(v_1)/(\lambda R_1),\cdots,g^{-1}(v_n)/(\lambda R_n))^T = S(x)$, $\mathrm{diag}[\lambda/(C_1(dg_1^{-1}(v_1)/dv_1)), \cdots ,\lambda/(C_n(dg_n^{-1}(v_n)/dv_n))] = H(x)$, $[T_{ij}]=T$ and

$(I_1,\cdots,I_n)^T=I$, then system (H') assumes the form of equation (I). Thus, system (H'), or equivalently system (H), is a special case of system (I).

Remark 3.2. In [3], Cohen and Grossberg establish some results for a class of competitive dynamical systems given by

$$x_i = a_i(x_i)\,[\,b_i(x_i) - \Sigma^n{}_{j=1}\, c_{ij}\, d_j(x_j)]\ ,i=1,\cdots,n. \qquad \text{(G)}$$

As pointed out in [2], if in (H), we let $1/C_i=a_i(u_i)$, $(-1/R_i)u_i + I_i=b_i(u_i)$, $T_{ij}= -c_{ij}$ and $g_j(\lambda u_j)=d_j(u_j)$, then system (H) is also a special case of (G). But it appears that the assumptions in [3] are such as to make the results of [3] not applicable to model (H).

We will study systems described by equation (I) under the following assumptions.

Assumption (A). For system (I) assume the following:

a) For every fixed $x\in(-1,1)^n$, $H(x)$ is symmetric and positive definite.

b) T is symmetric.

c) for each i, $1\le i\le n$, $s_i:(-1,1)\to\mathbf{R}$ is monotone-increasing, $s_i(0)=0$, $s_i(x_i)=-s_i(-x_i)$, the inverse $s_i^{-1}: \mathbf{R}\to(-1,1)$ exists, s_i and s_i^{-1} are C^1-functions, and furthermore, $s''_i(x_i)=d^2s_i(x_i)/dx_i^2$ exists.

Remark 3.3. The hypothesis concerning $H(x)$ in Assumption (A) for system (I) is weaker than the corresponding hypothesis made in [1] for the Hopfield model (H). The hypotheses concerning the matrix T and the functions s_i, $1\le i\le n$, enumerated in Assumption (A) for system (I) are identical to corresponding ones made in [1] for system (H), save that in assumption (A) it is assumed that s''_i exists for each i.

As in [1] and [3], we associate with system (I) an energy function $E : (-1,1)^n \to \mathbf{R}$, given by

$$E(x)= -(1/2)x^TTx + \Sigma^n{}_{i=1}\int_0^{x_i} s_i(\rho)\, d\rho - x^TI. \qquad (3.1)$$

We have that

$$\nabla E(x)= -Tx + S(x) - I,\quad J_E(x)= -T + \mathrm{diag}(\ s_1'(x_1),\cdots,s_n'(x_n)\ ) \text{ and}$$

$$D_{(I)}E(x) = \nabla E(x)^T(-H(x)(-Tx + S(x) - I)) = -\nabla E(x)^TH(x)\nabla E(x).$$

Thus the set of critical points of E is identical to the set of equilibrium points of system (I).

Assumption (B-1). Given assumption (A), we assume the following:

a) There is no $x\in(-1,1)^n$ satisfying the equations

(i) $\nabla E(x)=0$

(ii) $\det(J_E(x))=0$

(iii) $J_E(x)\ge 0$

(iv) $(s_1''(x_1),\cdots,s_n''(x_n))^T \perp N$, where $N=\{\ z=(y_1^3,\cdots,y_n^3)^T\in \mathbf{R}^n : J_E(x)y=0\}$

simultaneously.

b) The set of equilibrium points of (I) is discrete (i.e. each equilibrium of (I) is isolated).

Assumption (B-2). Given Assumption (A), we assume that there is no $x\in(-1,1)^n$ satisfying the equations

(i) $\nabla E(x)=0$

(ii) $\det(J_E(x))=0$

simultaneously.

Remark 3.4 Assumption (B-2) implies Assumption (B-1). Note, however, that Assumption (B-2) may be easier to apply than (B-1).

Remark3.5. If Assumption (A) is true for system (I) with T fixed, then Assumption (B-2) (and hence, Assumption (B-1)) will be true for almost all $I \in \mathbf{R}^n$ in the sense of the Lebesgue measure. Thus, any given Hopfield model (H) can be altered to satisfy Assumption (B-1) by changing the external input I by an arbitrarily small amount.

Assumption (C). Given Assumption (A), we assume that for each i, $s_i:(-1,1) \rightarrow \mathbf{R}$ is a C^2-function and $s''_i(\rho)>0$ for $\rho \in (0,1)$.

Remark 3.6. Again, this is not a strong assumption. For example, let us consider the specific case treated in [1], $s_i(\rho)=(2/(\lambda\pi R_i))\tan(\rho\pi/2)$. We have that $s'_i(\rho)=1/(\lambda R_i \cos^2(\rho\pi/2))$ and $s_i''(\rho)=\pi\sin(\rho\pi/2)/(\lambda R_i\cos^3(\rho\pi/2))$ which is continuous and $s_i''(\rho)>0$ for $\rho \in (0,1)$. So s_i satisfies the assumption (C).

IV. MAIN RESULTS: ANALYSIS

The results presented in this section for system (I) make use of Assumption (B-1). Since Assumption (B-2) implies Assumption (B-1), these results hold under (B-2) as well.

Theorem 4.1. If system (I) satisfies (A) and (B-1), then

1)for any $x \in (-1,1)^n$, there is a unique solution $\varphi(\cdot,0,x)$ for (I);

2)along a non-equilibrium solution of (I), energy function E given in (3.1) decreases monotonically. Thus, no non-constant periodic solutions exist;

3)each solution of (I) exits on $[0,+\infty)$;

4)each non-equilibrium solution of (I) converges to an equilibrium of (I) as $t \rightarrow +\infty$;

5) There are only finitely many equilibrium points for (I).

Theorem 4.2. Suppose that system (I) satisfies assumptions (A) and (B-1). If x is an equilibrium of (I) , then the following statements are equivalent

(i) x is stable;

(ii) x is a local minimum of function E;

(iii) $J_E(x)>0$;

(iv) x is asymptotically stable.

In the next result, we define 2^n regions in $(-1,1)^n$ by

$$\Lambda(\xi_1,\cdots,\xi_n) = \{ x \in (-1,1)^n : \xi_i x_i>0, \ 1 \le i \le n\} \tag{4.1}$$

where $\xi_i=1$ or -1, $1 \le i \le n$. For example, if n=2, we have $2^2 = 4$ regions $\Lambda(1,1)$, $\Lambda(1,-1)$, $\Lambda(-1,1)$ and $\Lambda(-1,-1)$ where ,e.g.,

$$\Lambda(1,-1) = \{x \in (-1,1)^2 : x_1>0, -x_2>0 \} = (0,1)\times(-1,0) ,$$

and so forth.

Theorem 4.3. If system (I) satisfies assumptions (A) and (B-1) and (C), then there is at most one (asymptotically) stable equilibrium of (I) in each of the 2^n regions $\Lambda(\xi_1,\cdots,\xi_n)$ defined by equation (4.1).

Corollary 4.1. If (I) satisfies assumptions (A) and (B-1) and (C) and if no (asymptotically) stable equilibrium of (I) has a 0- coordinate, then the total number of (asymptotically) stable equilibrium points of (I) is less or equal to 2^n (where n is the order of system (I)).

V. MAIN RESULTS: SYNTHESIS

Suppose that we are given information in the form of a collection of specified vectors which we wish to store as equilibrium points of a neural network of the form (I), satisfying Assumptions (A) and (C). In the present section we will address questions of the following type: (1) Can the specified information (vectors) be stored as equilibrium points of system (I) ? (2) If the answer to question (1) is affirmative, how can this be accomplished ? (3) Can this information (vectors) be stored as asymptotically stable equilibrium points of the neural network ?

In the following, we assume that the functions S(x) and H(x), which satisfy Assumptions (A) and (C), for system (I) are given and that m (information) vectors $a_1,\cdots,a_m$ in $(-1,1)^n$ are specified. We wish to determine an $n\times n$ symmetric matrix T and an external input vector $I\in R^n$ such that the vectors $a_1,\cdots,a_m$ are equilibrium points of system (I). In doing so, we touch on the questions raised above.

Lemma 5.1. Given m vectors $a_1,\cdots,a_m$ in $(-1,1)^n$, then the vectors $a_1,\cdots,a_m$ are equilibrium points of system (I), satisfying Assumptions (A) and (C), if and only if the unknown matrix Y=T and the unknown vector β=I comprise a solution of the system of equations given by

$$Ya_i + \beta = S(a_i) \quad , \quad i=1,\cdots,m$$

where the matrix $Y=[\eta_{ij}]=[\eta_{ji}]=Y^T$ contains $n(n+1)/2$ unknowns and $\beta=(\beta_1,\cdots,\beta_n)^T$ has n unknowns.

Lemma 5.2 Given m vectors $\{a_1,\cdots,a_m\}$ in $(-1,1)^n$, then Y=T and β=I comprise a solution of (5.1) if and only if Y=T and β=I are a solution of the equations

$$YA = B \ ,$$

$$\beta = Ya_m+S(a_m),$$

where $A=[a_1-a_m,\cdots,a_{m-1}-a_m]$, $B=[S(a_1)-S(a_m),\cdots,S(a_{m-1})-S(a_m)]$ and $Y=[\eta_{ij}] = =[\eta_{ji}]=Y^T$ contains $n(n+1)/2$ unknowns and $\beta=(\beta_1,\cdots,\beta_n)^T$ contains n unknowns.

Theorem 5.1. Given m vectors $a_1,\cdots,a_m$ in $(-1,1)^n$, then:

a) the vectors $a_1,\cdots,a_m$ are candidates for equilibrium points of system (I), satisfying Assumptions (A) and (C), if and only if

(i) the last (m-1-k) columns of U^TBV are zero;

(ii) A^TB is symmetric,

where $A=[a_1-a_m,\cdots,a_{m-1}-a_m]$, k=rank of A, and $B=[S(a_1)-S(a_m),\cdots,S(a_{m-1})-S(a_m)]$ and U and V are given in the singular value decomposition of A, $A=U\Sigma V^T$.

b) when part a) is true, we can synthesize a system (I) for which vectors $a_1,\cdots,a_m$ are equilibrium points, by determining the symmetric matrix T and the input vector I up to $(n-k)(n-k+1)/2$ unknown parameters, using the relation

$$T = UQU^T \quad \text{and} \quad I=S(a_m)-Ta_m$$

where

$$Q = \begin{pmatrix} C_1\Sigma_1^{-1} & (C_2\Sigma_1^{-1})^T \\ C_2\Sigma_1^{-1} & \Psi \end{pmatrix} , \tag{5.1}$$

where C_1 is formed by the first k rows and first k columns of U^TBV, C_2 is formed by the last (n-k) rows and first k columns of U^TBV, and Σ_1 is formed by the first k rows and first k columns of Σ.

Corollary 5.1. Given m linearly independent vectors $a_1,\cdots,a_m$ in $(-1,1)^n$, then:

a) the vectors $a_1,\cdots,a_m$ are candidates for equilibrium points of system (I), satisfying Assumptions (A) and (C), if and only if A^TB is symmetric;

b) when part a) is true, then the symmetric matrix T and vector I can be determined up to (n-m-1)(n-m)/2 unknown parameters.

Remark 5.1 1) The synthesis procedure of Theorem 5.1 will yield $a_1,\cdots,a_m$ as equilibrium points. But in general, system (I) constructed in this way, may possess other equilibrium points.

2) Once we have determined the matrix T and the vector I (by means of Theorem 5.1), we can use Theorem 4.2 or other results (cf. [7]) to determine which of the equilibrium points $a_1,\cdots,a_m$ are asymptotically stable. We have

Theorem 5.2. Suppose that system (I) has been synthesized by Theorem 5.1, with equilibrium points $a_1,\cdots,a_m$. Then a_i will be asymptotically stable if we can properly choose the (n-k)×(n-k) symmetric parameter sub-matrix Ψ in the matrix Q (see equation (5.1)) such that

$$\lambda_j<\alpha_i , \qquad j=1,\cdots,n$$

where λ_j, $1\le j\le n$, denote the eigenvalues of Q and $\alpha_i=\min\{s'_j(a_i^{(j)}) \}$, where $a_i^{(j)}$ is the j-th coordinate of a_i, $1\le j\le n$.

Remark 5.2. Theorem 5.2. gives a relation between the stability of a_i and the eigenvalues of the matrix Q. This suggests that we should determine the parameter matrix Ψ in such a way that the eigenvalues of Q are as small as possible. One easy way of doing this is to let $\Psi=-\alpha E$, where α is a large positive number and E is the identity matrix.

Remark 5.3. By Remark 3.5, Assumption (B-1) is true for system (I) for almost all choices of the matrix T and the vector I. When Assumption (B-1) is true, by Theorem 4.3, there is at most one asymptotically stable equilibrium of system (I) in each of the regions $\Lambda(\xi_1,\cdots,\xi_n)$ defined by equation (4.1). Thus, if there are $a_i\neq a_j$ in the given information vector set such that a_i and a_j are in the same region $\Lambda(\xi_1,\cdots,\xi_n)$, then it may not be possible to store a_i or a_j as asymptotically stable equilibrium points of system (I).

VI. SYNTHESIS PROCEDURE AND AN EXAMPLE

In the present section, we summarize the the results of section 5 and we demonstrate the applicability of our results by constructing a three dimensional neural network with specific (asymptotically stable) equilibrium points.

We assume that the functions $S:(-1,1)^n\rightarrow R^n$ and $H:(-1,1)\rightarrow R^{n\times n}$ are given and satisfy Assumptions (A) and (C). Also, we assume that we are given m information vectors $a_1,\cdots,a_m$ in R^n which are to be stored as (asymptotically stable) equilibrium points in a neural network of the form (I). Our goal, then, is to determine an n×n symmetric matrix T and an n-dimensional vector I such that the (asymptotically stable) equilibrium points of

$$\dot{x} = -H(x)(-Tx + S(x) - I) \qquad \text{(I)}$$

include the vectors $a_1,\cdots,a_m$.

Summary of Procedure:

i) Check if there are $a_i \neq a_j$ which are located in the same region $\Lambda(\xi_1,\cdots,\xi_n)$ defined by equation (4.1). If this is true, it may not be possible to store a_i or a_j as asymptotically stable equilibrium points of (I);

ii) Compute $A=[a_1-a_m,\cdots,a_{m-1}-a_m]$,
$S(a_i)$, $i=1,\cdots,m$,
$B=[S(a_1)-S(a_m),\cdots,S(a_{m-1})-S(a_m)]$, and
$A^T B$;

iii) Check if $A^T B$ is symmetric. If this is not true, $a_1,\cdots,a_m$ can not all be equilibrium points of (I);

iv) Perform a singular value decomposition of A to obtain the matrices U, V and Σ such that $A=U\Sigma V^T$, where U, V are unitary matrices and where Σ is a diagonal matrix with the singular values of A on its diagonal. (This can be accomplished by standard computer routines.) In doing so, we determine k = the rank of A which is equal to the number of non-zero diagonal elements of Σ;

v) Compute $U^T BV$;

vi) Check if the last (m-1-k) columns of $U^T BV$ are all zero. If this is not true, then $a_1,\cdots,a_m$ can not all be equilibrium points of (I);

vii) Choose an $(n-k)\times(n-k)$ symmetric parameter matrix Ψ. We may choose $\Psi = -\alpha E$, where α is a large positive real number and E is the $(n-k)\times(n-k)$ identity matrix;

viii) Compute

$$Q = \begin{pmatrix} C_1\Sigma_1^{-1} & (C_2\Sigma_1^{-1})^T \\ C_2\Sigma_1^{-1} & \Psi \end{pmatrix},$$

where C_1 is formed by the first k rows and first k columns of $U^T BV$, C_2 is formed by the last (n-k) rows and first k columns of $U^T BV$, and Σ_1 is formed by the first k rows and first k columns of Σ;

ix) Compute
$$T = U \cdot QU^T \quad \text{and}$$
$$I = S(a_m) - T(a_m);$$

x) Check if $S(a_i)-T(a_i)-I=0$, $i=1,\cdots,m$. This should be true and $a_1,\cdots,a_m$ are equilibrium points of system (I);

xi) Check if all of the eigenvalues of $J_E(a_i)=\mathrm{diag}[s'_1(a_i^{(1)}),\cdots,s'_n(a_i^{(n)})]-T$ are positive, where $a_i^{(j)}$ is the j-th coordinate of the vector a_i, $i=1,\cdots,m$. If this is true, then $a_1,\cdots,a_m$ are asymptotically stable equilibrium points of (I).

Example 6.1. Let n=3 and let function s: (-1,1)→**R** be given by

$$s(\rho)=(1/\lambda)(2/\pi)\tan((\pi/2)\rho)$$

where $\lambda=1.4$. Then

$$s'(\rho)=(1/\lambda)[1/\cos^2((\pi/2)\rho)], \quad \text{and}$$
$$s''(\rho)=(\pi/\lambda)\sin((\pi/2)\rho)/\cos^3((\pi/2)\rho).$$

Let $S:(-1,1)^3 \to R^3$ be given by $S(x)=(s(x_1),s(x_2),s(x_3))^T$ and let $H:(-1,1)^3 \to R^{3\times 3}$ be given by $H(x)=\mathrm{diag}[1/s'(x_1),1/s'(x_2),1/s'(x_3)]$. Then both S and H satisfy Assumptions (A) and (C).

Also, let m=2 and let the desired equilibrium points for (I) be given by
$a_1=(-0.6, 0.5,-0.4)^T$ and $a_2=(0.4, 0.8,-0.7)^T$.

We aim to find a 3×3 symmetric matrix T and a 3-dimensional vector I for a neural network of the form (I), having asymptotically stable equilibrium points a_1 and a_2.

In the following, we use the procedure given above, employing identical steps in our procedure:

i) Since $-0.6<0$ and $0.4>0$, a_1 and a_2 are not in the same region $\Lambda(\xi_1,\xi_2,\xi_3)$. Accordingly, it may be possible to store a_1 and a_2 as asymptotically stable equilibrium points of (I);

ii) We compute A,B, and A^TB as

$$A=\begin{pmatrix} -1.0000E+00 \\ -3.0000E-01 \\ 3.0000E-01 \end{pmatrix},$$

$$S(a_1)=(-6.2588E-01,4.5473E-01,-3.3038E-01)^T,$$

$$S(a_2)=(3.3038E-01,1.3995E+00,-8.9245E-01)^T,$$

$$B=\begin{pmatrix} -9.5626E-01 \\ -9.4478E-01 \\ 5.6028E-01 \end{pmatrix}, \qquad \text{and}$$

$$A^TB= [\ 1.4083E+00\ ]\ ;$$

iii) Since A^TB is a 1×1 matrix, it is symmetric. Accordingly, a_1 and a_2 may be equilibrium points of system (I);

iv) A singular value decomposition of A, $A=U\Sigma V^T$, yields

$$U=\begin{pmatrix} -9.2057E-01 & -2.7617E-01 & 2.7617E-01 \\ -2.7617E-01 & 9.6029E-01 & 3.9713E-02 \\ 2.7617E-01 & 3.9713E-02 & 9.6029E-01 \end{pmatrix},$$

$$\Sigma=\begin{pmatrix} 1.0863E+00 \\ 0 \\ 0 \end{pmatrix},$$

$$V= [\ 1.0\ ],$$

and the rank of A is equal to k = 1;

v) Compute the matrix U^TBV as

$$U^TBV=\begin{pmatrix} 1.2965E+00 \\ -6.2085E-01 \\ 2.3814E-01 \end{pmatrix};$$

vi) Since m-1-k=2-1-1=0, no columns of U^TBV need to be checked. Accordingly, a_1 and a_2 can be equilibrium points of (I);

vii) Choose the parameter matrix Ψ as

$$\Psi = -\alpha \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

where $\alpha = 10$;

viii) Compute the matrix Q as

$$Q = \begin{pmatrix} 1.1935E+00 & -5.7154E-01 & 2.1923E-01 \\ -5.7154E-01 & -1.0000E+01 & 0 \\ 2.1923E-01 & 0 & -1.0000E+01 \end{pmatrix};$$

ix) Compute T and I as

$$T = \begin{pmatrix} -9.1608E-01 & 3.2827E+00 & -2.9584E+00 \\ 3.2827E+00 & -8.8479E+00 & -1.0548E+00 \\ -2.9284E+00 & -1.0548E+00 & -9.0425E+00 \end{pmatrix},$$

and

$$I = \begin{pmatrix} -4.0002E+00 \\ 6.4264E+00 \\ -5.1950E+00 \end{pmatrix};$$

x) Since

$$S(a_1)-Ta_1-I = \begin{pmatrix} 0 \\ 2.2204E-16 \\ 0 \end{pmatrix}$$

and

$$S(a_2)-Ta_2-I = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

a_1 and a_2 are equilibrium points of system (I);

xi) Since the eigenvalues of $J_E(a_1)$,

6.8059E-01, 1.1163E+01, 1.1550E+01

are all positive, and since the eigenvalues of $J_E(a_2)$,

4.8305E-01, 1.3276E+01, 1.7085E+01

are all positive, the vectors a_1 and a_2 are asymptotically stable equilibrium points of system (I).

VII. CONCLUDING REMARKS

In this paper, we considered a class of neural networks which includes the Hopfield model. For this class of systems we established a variety of results which enable us to perform a qualitative analysis of such systems. For this class of systems we also developed results which enable us to synthesize neural networks with specified asymptotically stable equilibrium points. Work in progress aims to remove some of the limitations of our analysis and synthesis procedures.

Most (but not all) of the present results are global. In a companion paper, [7], we concentrate on local results.

REFERENCES

[1] J.J. Hopfield, " Neurons with graded response have collective computational properties like those of two-state neurons", Proc. Natl. Acad. Sci. USA Vol. 81, May 1984, pp. 3088-3092.

[2] G.A. Carpenter, M. Cohen and S. Grossberg, "Computing with Neural Networks", Science, Vol. 235, 1987, pp. 1226-1227.

[3] M. Cohen and S. Grossberg, "Absolute Stabilty of Global Pattern Formation and Parallel Memory Storage By Competitive Neural Networks", IEEE Trans. Syst. Man Cybern. SMC-13, 1983, pp. 815-826.

[4] R.K. Miller and A.N. Michel, Ordinary Differential Equitions, Academic Press, New York,1982.

[5] H.L.Royden, Real Analysis, Second Edition, The Macmillan Co., New York, 1963.

[6] A. Avez, Differential Calculus, John Wiley and Sons, New York, 1986.

[7] A.N. Michel, J.A. Farrell and W.Porod, " Qualitative Analysis of Neural Networks", Submitted for publication.

QUALITATIVE ANALYSIS OF NEURAL NETWORKS

A. N. MICHEL, J. A. FARRELL, and W. POROD
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

I. INTRODUCTION

In the present paper we will address the following problems: a) Determine the stability properties of a given equilibrium point of a neural network. b) Given that a specific equilibrium point of a neural network is asymptotically stable, establish an estimate for its domain of attraction. c) Given a set of initial conditions and external inputs, establish estimates for corresponding trajectory bounds.

In arriving at the results of the present paper, we make use of the method of analysis advanced in [8]. Specifically, we view a high dimensional neural network as an interconnection of individual neurons (subsystems). Our results are phrased in terms of the qualitative properties of the free subsystems (individual neurons, disconnected from the network) and in terms of the properties of the interconnecting structure of the neural network. As such, these results constitute useful design tools. This approach makes possible the systematic analysis of high dimensional complex systems and it frequently enables one to circumvent difficulties encountered in the analysis of such systems by conventional methods.

II. INTERCONNECTED SYSTEMS: NEURAL NETWORKS

The present section consists of two parts. First, we present the model of neural networks which we consider. Next, we formulate such networks as interconnected systems (i.e., as an interconnection of several single neurons or subsystems.

A. The Hopfield Model for Neural Networks

In the present paper we consider neural networks of the Hopfield type [1] as shown in Figure 1. Such systems can be represented by equations of the form

$$\dot{u}_i = \sum_{j=1}^{N} A_{ij}v_j - b_iu_i + U_i(t), \text{ for } i = 1,\dots,N \tag{I}$$

where $A_{ij} = \dfrac{T_{ij}}{C_i}$, $U_i(t) = \dfrac{I_i(t)}{C_i}$ and $b_i = \dfrac{1}{\tau_i C_i}$, $C_i > 0$, $T_{ij} = \dfrac{1}{R_{ij}}$,

$R_{ij} \in R = (-\infty, \infty)$, $\dfrac{1}{\tau_i} = \dfrac{1}{R_i} + \sum_{j=1}^{N} |T_{ij}|$, $R_i > 0$, $U_i : R^+ = [0, \infty) \to R$, U_i is continuous, $\dot{u}_i = \dfrac{du_i}{dt}$, $v_i = G_i(u_i)$, $G_i : R \to (c_1, c_2) \subset (-2,2)$ (where $c_1 < 0 < c_2$), G_i is continuously differentiable and strictly monotonically increasing, $u_iG_i(u_i) > 0$ for all $u_i \neq 0$, and $G_i(0) = 0$. In (I), C_i denotes capacitance, R_{ij} denotes resistance (possibly including a sign inversion due to an amplifier), $G_i(\cdot)$ denotes an amplifier nonlinearity, and $U_i(\cdot)$ denotes an external input.

In the literature it is frequently assumed that $T_{ij} = T_{ji}$ for all $i, j = 1,\dots,N$ and that $T_{ii} = 0$ for all $i = 1,\dots,N$. We will make these assumptions only when explicitly stated.

---

*The work of A. N. Michel and J. A. Farrell was supported by the National Science Foundation under grant ECS84-19918.

**The work of W. Porod was supported by SDIO/IST and managed by the ONR under grant N00014-86-K-0506.

We are interested in the qualitative behavior of solutions of (I) near equilibrium points (rest positions where $u_i \equiv 0$, $i = 1,\dots,N$). By setting the external inputs $U_i(t)$, $i = 1,\dots,N$, equal to zero, we define $u^* = [u_1^*,\dots,u_N^*]^T \in R^N$ to be an equilibrium for (I) provided that

$$-b_i u_i^* + \sum_{j=1}^{N} A_{ij}G_j(u_j^*) = 0, \text{ for } i = 1,\dots,N. \qquad (1)$$

The locations of such equilibria in R^N are determined by the interconnection pattern of the neural network (i.e., by the parameters A_{ij}, $i,j = 1,\dots,N$) as well as by the parameters b_i and the nature of the nonlinearities $G_i(\cdot)$, $i = 1,2,\dots,N$.

Throughout, we will assume that a given equilibrium u^* being analyzed is an isolated equilibrium for (I). Also when analyzing the stability properties of a given equilibrium point, we will be able to assume, without loss of generality, that this equilibrium is located at the origin of R^N.

B. Interconnected Systems

We will find it convenient to view system (I) as an interconnection of N free subsystems described by equations of the form

$$\dot{p}_i = -b_i p_i + A_{ii}G_i(p_i) + U_i(t). \qquad (S_i)$$

Under this viewpoint, the terms

$$g_i(x_1,\dots,x_n) \overset{\Delta}{=} \sum_{\substack{j=1 \\ i\neq j}}^{N} A_{ij}G_j(x_j), \text{ for } i = 1,\dots,N \qquad (2)$$

make up the interconnecting structure of the system (I).

Following the method of analysis advanced in [8], we will establish stability results for system (I) which are phrased in terms of the qualitative properties of the free subsystems (S_i) and in terms of the properties of the interconnecting structure given in (2). This method of analysis makes it often possible to circumvent difficulties that arise in the analysis of complex high-dimensional systems. Furthermore, results obtained in this manner frequently yield insight into the dynamic behavior of systems in terms of system components and interconnections.

III. ANALYSIS OF NEURAL NETWORKS (INTERCONNECTED SYSTEMS)

The present section is divided into four parts. In the first two parts we present results for the asymptotic and exponential stability of equilibria in neural networks (I). In the third part, we establish estimates of trajectory bounds for (I) while in the fourth part we determine estimates of the domain of attraction of an asymptotically stable equilibrium for (I).

A. General Stability Conditions

The principal Lyapunov stability results for systems are presented, e.g., in Chapter 5 of [9].

We will utilize the following hypotheses in our first result.

(A-1) For system (I), the external inputs are all zero, i.e.,

$$U_i(t) \equiv 0, \ i = 1,\dots,N.$$

(A-2) For system (I), the interconnections satisfy the estimate

$$x_i A_{ij} G_j(x_j) \leq x_i a_{ij} x_j$$

for all $|x_i| < r_i$, $|x_j| < r_j$, $i,j = 1,\ldots,N$, where the a_{ij} are real constants.

(A-3) There exists an N-vector $\alpha > 0$ (i.e., $\alpha^T = (\alpha_1,\ldots,\alpha_N)$ and $\alpha_i > 0$, $i = 1,\ldots,N$) such that the *test matrix* $S = [s_{ij}]$

$$s_{ij} = \begin{cases} \alpha_i(-b_i + a_{ii}), & i = j \\ (\alpha_i a_{ij} + \alpha_j a_{ji})/2, & i \neq j \end{cases}$$

is negative definite, where the b_i are defined in (I) and the a_{ij} are given in (A-2).

We are now in a position to state and prove the following result.

Theorem 1. The equilibrium $x = 0$ of the neural network (I) is *exponentially stable* if hypotheses (A-1), (A-2) and (A-3) are satisfied.

Proof. For (I) we choose the Lyapunov function

$$v(x) = \sum_{i=1}^{N} \frac{1}{2} \alpha_i x_i^2 \tag{3}$$

where the α_i are given in (A-3). This function is clearly positive definite. The time derivative of v along the solutions of (I) is given by

$$Dv_{(I)}(x) = \sum_{i=1}^{N} \frac{1}{2} \alpha_i(2x_i)\left[-b_i x_i + \sum_{j=1}^{N} A_{ij} G_j(x_j)\right]$$

where (A-1) has been invoked. In view of (A-2) we have

$$\begin{aligned} Dv_{(I)}(x) &\leq \sum_{i=1}^{N} \alpha_i\left(-b_i x_i^2 + x_i \sum_{j=1}^{N} a_{ij} x_j\right) \\ &= x^T R x \qquad \text{for all } |x|_2 < r \end{aligned}$$

where $r = \min_i(r_i)$, $|x|_2 = \left(\sum_{i=1}^{N} x_i^2\right)^{1/2}$, and $R = [r_{ij}]$ is given by

$$r_{ij} = \begin{cases} \alpha_i(-b_i + a_{ii}), & i = j \\ \alpha_i a_{ij}, & i \neq j \end{cases} .$$

But

$$x^T R x = x^T\left(\frac{R + R^T}{2}\right)x = x^T S x \leq \lambda_M(S)|x|_2^2 \tag{4}$$

where S is the matrix given in (A-3) and $\lambda_M(S)$ denotes the largest eigenvalue of the real symmetric matrix S. Since S is by assumption negative definite, we have $\lambda_M(S) < 0$. It follows from (3) and (4) that in some neighborhood of the origin $x = 0$, we have

$$c_1|x|_2^2 \leq v(x) \leq c_2|x|_2^2, \quad Dv_{(I)}(x) \leq -c_3|x|_2^2$$

where $c_1 = \frac{1}{2}\min_i \alpha_i > 0$, $c_2 = \frac{1}{2}\max_i \alpha_i > 0$, and $c_3 = -\lambda_M(S) > 0$. Hence, the equilibrium $x = 0$ of the neural network (I) is exponentially stable (c.f. Theorem 9.10 in [9]).

Remark 1. Consistent with the philosophy of viewing the neural network (I) as an interconnection of N free subsystems (S_i), we think of the Lyapunov function (3) as consisting of a weighted sum of Lyapunov functions for each free subsystem (S_i) (with $U_i(t) \equiv 0$). There are two reasons for using the weighting vector $\alpha > 0$. First, it provides flexibility to emphasize the relative importance of the qualitative properties of the various individual subsystems. Secondly, as will be seen in the next subsection, it provides a vehicle for establishing additional and simpler results.

Remark 2. It is emphasized that Theorem 1 does not require that the parameter A_{ij} in (I) form a symmetric matrix. In other words, the parameters T_{ij} need not form a symmetric matrix.

Remark 3. Hypothesis (A-2) provides a measure of interaction between the various subsystems (S_i).

In order to satisfy hypothesis (A-3), it is necessary that

$$-b_i + a_{ii} < 0, \quad i = 1,\ldots,N. \tag{5}$$

Now if we let $i = j$ in (A-2), we have

$$A_{ii}x_iG_i(x_i) \leq a_{ii}x_i^2$$

and if we apply the sector conditions (A-4), given in the next subsection, we obtain

$$a_{ii} = A_{ii}\delta_i$$

where

$$\delta_i = \begin{cases} \sigma_{i2} \text{ when } A_{ii} > 0 \\ \sigma_{i1} \text{ when } A_{ii} < 0. \end{cases}$$

If we choose for the free subsystem (S_i) (with $U_i(t) \equiv 0$) the Lyapunov function $v_i(p_i) = \frac{1}{2}\,p_i^2$, then we obtain under the above assumptions the estimate

$$Dv_i(p_i) \leq \sigma_i p_i^2 \text{ , where } \sigma_i = b_i + A_{ii}\delta_i.$$

This shows that the term $\sigma_i < 0$ constitutes a measure of the degree of stability for the free subsystem (S_i). The same reasoning shows that collectively, the numbers σ_i, $i = 1,\ldots,N$, will have stabilizing effects on the overall neural network while the sign-sensitive off-diagonal terms in the matrix S in (A-3) may have beneficial stabilizing effects or destabilizing effects. Thus, in order to satisfy (5), and hence (A-3), it is necessary that the equilibrium $p_i = 0$ of each free subsystem (S_i) (with $U_i(t) \equiv 0$) is exponentially stable.

B. Special Cases: Weak-Coupling Conditions

The test matrix S given in hypothesis (A-3) has off-diagonal terms which may be positive or nonpositive (i.e., they are sign-sensitive). For the special case where the off-diagonal terms of the test matrix $S = [s_{ij}]$ are non-negative (see (A-4), (A-5)), equivalent stability results may be obtained which are much easier to apply than, e.g., Theorem 1. Such results are called weak-coupling conditions in the literature [8], [10] since they involve sign insensitive hypotheses for the interconnections of system (I). The condition $s_{ij} \geq 0$ for all $i \neq j$ may reflect properties of the system (I) (i.e., it may be true that $A_{ij} \geq 0$ for all $i \neq j$) or they may be the consequence of a majorization process. In the proofs of several subsequent results, we will make use of some of the properties of M-matrices (see, e.g., [8, Chapter II]). In addition we will use the following assumptions.

(A-4) For system (I) the nonlinearlity $G_i(x_i)$ satisfies the sector condition

$$0 < \sigma_{i1} \leq \frac{G_i(x_i)}{x_i} \leq \sigma_{i2}, \text{ for all } |x_i| < r_i,\ i = 1,\ldots,N.$$

(A-5) The successive principal minors of the N×N test matrix $D = [d_{ij}]$ are all positive, where

$$d_{ij} = \begin{cases} \dfrac{b_i}{\sigma_{i2}} - A_{ii}, & i = j \\ -|A_{ij}|, & i \neq j \end{cases}$$

where the b_i and A_{ij} are defined in (I) and σ_{i2} is defined in (A-4).

Theorem 2. The equilibrium $x = 0$ of the neural network (I) is asymptotically stable if hypotheses (A-1), (A-4) and (A-5) are true.

Proof. For (I) we choose the Lyapunov function

$$v(x) = \sum_{i=1}^{N} \alpha_i |x_i|. \tag{6}$$

Along the solutions of (I) we have

$$Dv_{(I)}(x) \leq \sum_{i=1}^{N} \alpha_i \{-b_i|x_i| + \frac{A_{ii}G_i(x_i)}{x_i}|x_i| + \sum_{\substack{j=1\\ i\neq j}}^{N} |A_{ij}| \frac{G_j(x_j)}{x_j}|x_j|\}$$

$$\leq \sum_{i=1}^{N} \alpha_i \{[-(\frac{b_i}{\sigma_{i2}}) + A_{ii}] \frac{G_i(x_i)}{x_i}|x_i| + \sum_{\substack{j=1\\ i\neq j}}^{N} |A_{ij}| \frac{G_j(x_j)}{x_j}|x_j|\}$$

$$= -\alpha^T D w$$

where $w^T = [\frac{G_1(x_1)}{x_1}|x_1|,\ldots,\frac{G_N(x_N)}{x_N}|x_N|] = (w_1,\ldots,w_N)$, and D is the test matrix in hypothesis (A-5). Then, $w = 0$ if and only if $x = 0$ and $w_i > 0$, $i = 1,\ldots,N$ when $x \neq 0$. Since $d_{ij} \leq 0$ for all $i \neq j$, and since by assumption the successive principal minors of D are positive, it follows that D is an M-matrix. Because D is an M-matrix, D^{-1} exists and $D^{-1} \geq 0$ (i.e., $[D^{-1}]_{ij} \geq 0$ for all $i,j = 1,\ldots,N$). If we let

$$y^T = \alpha^T D,$$

then

$$\alpha = (D^{-1})^T y. \tag{7}$$

Now choose a vector $y > 0$ (i.e., $y_i > 0$, $i = 1,\ldots,N$). Then $\alpha > 0$ and $v(x)$ is positive definite and

$$Dv_{(I)}(x) \leq -y^T w$$

is negative definite in some neighborhood of the origin x = 0. Hence, the equilibrium x = 0 of system (I) is asymptotically stable (see Theorem 2.2.23 in [8]).

In other words, the above argument shows that if the hypotheses of Theorem 2 are satisfied, then the Lyapunov function v(x) given in (6), which may be viewed as a generalized Hamming distance of the state vector x from the origin 0, will decrease with time and approach the origin as $t \to \infty$.

C. Estimates of Trajectory Bounds

In general, one is not only interested in questions concerning the stability of an equilibrium of the system (I), but also in performance. One way of assessing the qualitative properties of the neural system (I) is by investigating solution bounds near an equilibrium of interest. We present here such a result by assuming that the hypotheses of Theorem 2 are satisfied.

In the following, we will not require that the external inputs $U_i(t)$, $i = 1,\ldots,N$ be zero. However, we will need to make the additional assumptions enumerated below.

(A-6) Assume that there exist $\lambda_i > 0$, $i = 1,\ldots,N$, and an $\varepsilon > 0$ such that

$$\frac{b_i}{\sigma_{i2}} - A_{ii} - \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{\lambda_j}{\lambda_i} |A_{ji}| \geq \varepsilon > 0, \quad i = 1,\ldots,N \tag{8}$$

where b_i and A_{ij} are defined in (I) and σ_{i2} is defined in (A-4). This assumption is a minor modification of a property of M-matrices.

(A-7) Assume that for system (I),

$$\sum_{i=1}^{N} \lambda_i |U_i(t)| \leq k \text{ for all } t \geq 0 \tag{9}$$

for some constant $k > 0$ where the λ_i, $i = 1,\ldots,N$ are defined in (A-6).

In the proof of our next theorem, we will make use of a comparison result. We consider a scalar comparison equation of the form

$$\dot{y} = G(y) \qquad (C)$$

where $y \in R$, $G : B(r) \to R$ for some $r > 0$, and G is continuous on $B(r) = \{x \in R : |x| < r\}$.

Theorem A. Let p(t) denote the maximal solution of (C) with $p(t_0) = y_0 \in B(r)$, $t \geq t_0 > 0$. If r(t), $t \geq t_0 \geq 0$ is a continuous function such that $r(t_0) \leq y_0$, if r(t) satisfies the differential inequality

$$Dr(t) = \lim_{k \to 0^+} \frac{1}{k} \sup[r(t+k) - r(t)] \leq G(r(t))$$

almost everywhere, then

$$r(t) \leq p(t) \text{ for } t \geq t_0 \geq 0,$$

for as long as both r(t) and p(t) exist.

For the proof of the above result, as well as other comparison theorems, see, e.g., [8] and [9].

Notation. In the following, we let

$$\delta = \min_i \sigma_{i1} \tag{10}$$

where σ_{i1} is defined in (A-4), we let

$$c = \varepsilon\delta, \tag{11}$$

where ε is given in (A-6), and we let

$$\phi(t,t_0,x_0) = [\phi_1(t,t_0,x_0),\dots,\phi_N(t,t_0,x_0)]^T$$

denote the solution of (I) with $\phi(t_0,t_0,x_0) = x_0 = (x_{10},\dots,x_{N0})^T$ for some $t_0 \geq 0$.

We are now in a position to prove the following result.

Theorem 3. Assume that hypotheses (A-6) and (A-7) are satisfied. Then

$$||\phi(t,t_0,x_0)|| \overset{\Delta}{=} \sum_{i=1}^{N} \lambda_i|\phi_i(t,t_0,x_0)| \leq \left(\alpha - \frac{k}{c}\right)e^{-c(t-t_0)} + \frac{k}{c},\ t \geq t_0 \geq 0 \tag{12}$$

provided that $\alpha > k/c$ and $||x_0|| = \sum_{i=1}^{N} \lambda_i|x_{i0}| \leq \alpha$, where the λ_i, $i = 1,\dots,N$ are given in (A-6) and k is given in (A-7).

Proof. For (I) we choose the Lyapunov function

$$v(x) = \sum_{i=1}^{N} \lambda_i|x_i|. \tag{13}$$

Along the solutions of (I) we obtain

$$Dv_{(I)}(x) \leq \lambda^T Dw + \sum_{i=1}^{N} \lambda_i|U_i(t)| \tag{14}$$

where $w^T = \left[\frac{G_1(x_1)}{x_1}|x_1|,\dots,\frac{G_N(x_N)}{x_N}|x_N|\right]$, $\lambda = (\lambda_1,\dots,\lambda_N)^T$, and $D = [d_{ij}]$ is the test matrix given in (A-5). Note that when (A-6) is satisfied, as in the present theorem, then (A-5) is automatically satisfied. Note also that $w \geq 0$ (i.e., $w_i \geq 0$, $i = 1,\dots,N$) and $w = 0$ if and only if $x = 0$.

Using manipulations involving (8) - (11) and (14), it is easy to show that

$$Dv_{(I)}(x) \leq -cv(x) + k.$$

The above inequality yields now the comparison equation

$$\dot{y} = -cy + k. \qquad (C')$$

The unique solution of (C') is given by

$$p(t,t_0,p_0) = \left(p_0 - \frac{k}{c}\right)e^{-c(t-t_0)} + \frac{k}{c},\ t \geq t_0,$$

where the notation of Theorem A has been used. If we let r = v, then we obtain from Theorem A that

$$p(t) \geq r(t) = v(\phi(t,t_0,x_0)) = \sum_{i=1}^{N} \lambda_i |\phi_i(t,t_0,x_0)| = ||\phi(t,t_0,x_0)||,$$

i.e., the desired estimate given by (12) is true, provided that $|r(t_0)| = \sum_{i=1}^{N} \lambda_i |x_{i0}| = ||x_0|| \leq \alpha$ and $\alpha > k/c$.

We conclude the present subsection by noting that methods for determining the optimal choice of the λ_i in Theorem 3, to obtain the least conservative estimate in (12), do not seem to have been determined at this time.

D. Estimates of the Domain of Attraction of an Equilibrium

Neural networks of the type considered herein have many equilibrium points. If a given equilibrium is asymptotically stable, or exponentially stable, then the extent of this stability is of interest. As usual, we assume that $x = 0$ is the equilibrium of interest. If $\phi(t,t_0,x_0)$ denotes a solution of the network (I) with $\phi(t_0,t_0,x_0) = x_0$, then we would like to know for which points x_0 it is true that $\phi(t,t_0,x_0)$ tends to the origin as $t \to \infty$. The set of all such points x_0 makes up the domain of attraction (the basin of attraction) of the equilibrium $x = 0$. In general, one cannot determine such a domain in its entirety. However, several techniques have been devised to estimate subsets of a domain of attraction. We apply one such method to neural networks, making use, e.g., of Theorem 1. This technique is applicable to our other results as well, by making obvious modifications.

We assume that the hypotheses (A-1), (A-2) and (A-3) are satisfied and for the free subsystem (S_i) we choose the Lyapunov function

$$v_i(p_i) = \frac{1}{2} p_i^2.$$

Then $Dv_{i_{(S_i)}}(p_i) \leq (-b_i+a_{ii})p_i^2$, $|p_i| < r_i$ for some $r_i > 0$. If (A-3) is satisfied, we must have $(-b_i+a_{ii}) < 0$ and $Dv_{i_{(S_i)}}(p_i)$ is negative definite over $B(r_i)$.

Let $C_{v_{0i}} = \{p_i \in R : v_i(p_i) = \frac{1}{2} p_i^2 < \frac{1}{2} r_i^2 \overset{\Delta}{=} v_{0i}\}$. Then $C_{v_{0i}}$ is contained in the domain of attraction of the equilibrium $p_i = 0$ for (S_i).

To obtain an estimate for the domain of attraction of $x = 0$ for the neural network (I), we use the Lyapunov function

$$v(x) = \sum_{i=1}^{N} \frac{1}{2} \alpha_i x_i^2 = \sum_{i=1}^{N} \alpha_i v_i(x_i).$$

It is now an easy matter to show that the set

$$C_\lambda = \{x \in R^N : v(x) = \sum_{i=1}^{N} \alpha_i v_i(x_i) < \lambda\}$$

will be a subset of the domain of attraction of $x = 0$ for the neural network (I), where

$$\lambda = \min_{1 \leq i \leq N} (\alpha_i v_{0i}) = \min_{1 \leq i \leq N} \left(\frac{1}{2} \alpha_i r_i^2\right).$$

In order to obtain the best estimate of the domain of attraction of $x = 0$ by the present method, we must choose the α_i in an optimal fashion. The reader is referred to [10], [11], [12] where several methods to accomplish this are discussed.

REFERENCES

[1] J. J. Hopfield and D. W. Tank, "Computing with Neural Circuits: A Model," Science, vol. 233, 1986, pp. 625-633.

[2] Neural Networks for Computing, John S. Denker, Editor, American Institute of Physics Conference 151, Snowbird, Utah, 1986.

[3] G. E. Hinton and J. A. Anderson, Editors, Parallel Models of Associative Memory, Erlbaum, 1981.

[4] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, 1984.

[5] J. J. Hopfield, "Neural Networks and Physical Systems with emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. U.S.A., vol. 79, 1982, pp. 2554-2558.

[6] J. J. Hopfield, "Neurons with Graded Response have Collective Computational Properties like those of two-state Neurons," Proc. Natl. Acad. Sci. U.S.A., vol. 81, 1984, pp. 3088-3092.

[7] G. A. Carpenter, M. A. Cohen, and S. Grossberg," Computing with Neural Networks," Science, Vol. 235, 1987, pp. 1226-1227.

[8] A. N. Michel and R. K. Miller, Qualitative Analysis of Large Scale Dynamical Systems, Academic Press, 1977.

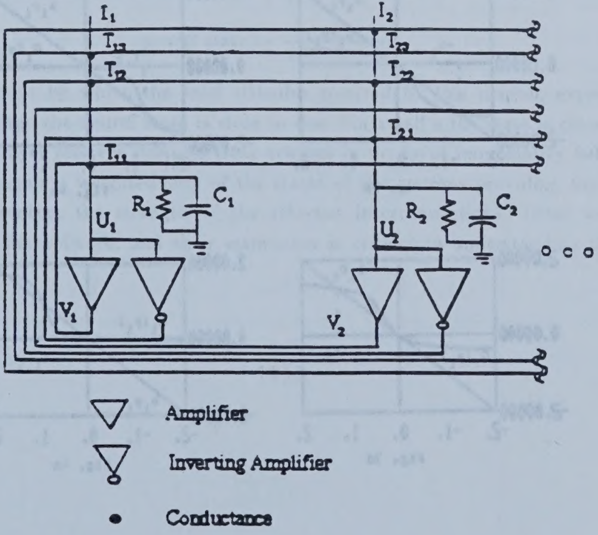
[9] R. K. Miller and A. N. Michel, Ordinary Differential Equations, Academic Press, 1982.

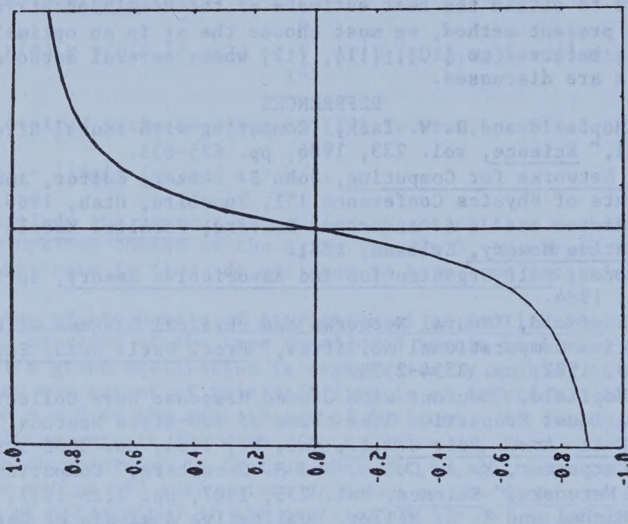
[10] A. N. Michel, "On the Status of Stability of Interconnected Systems," IEEE Transactions on Automatic Control, vol. 28, no. 6, June 1983, pp. 639-653.

[11] M. A. Pai, Power System Stability, Amsterdam, The Netherlands: North Holland, 1981.

[12] A. N. Michel, N. R. Sarabudla, and R. K. Miller, "Stability Analysis of Complex Dynamical Systems: Some Computational Methods," Circuits, Systems and Signal Processing, vol. 1, no. 2, 1982, pp. 171-202.

[13] I. W. Sandberg, "Some Theorems on Dynamic Response of Nonlinear Transistor Networks," Bell System Tech. J., vol. 48, 1969, pp. 35-54.





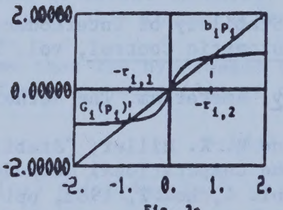


Fig. 3a

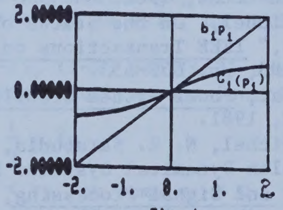


Fig. 4a

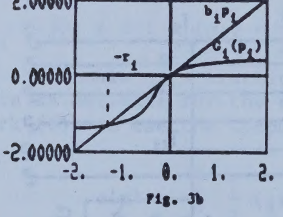


Fig. 3b

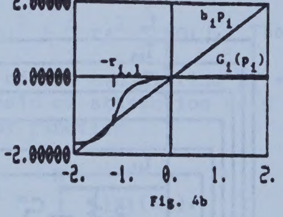


Fig. 4b

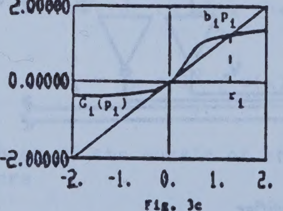


Fig. 3c

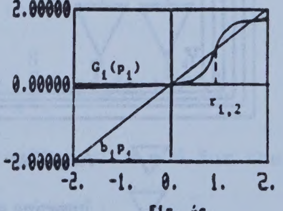


Fig. 4c

CONVERGENCE ISSUES IN PERCEPTRON BASED ADAPTIVE NEURAL NETWORK MODELS

by YASH SHRIVASTAVA and SOURA DASGUPTA
Department of Electrical and Computer Engineering
University of Iowa
Iowa City, Iowa 52242

ABSTRACT

This paper considers the convergence behavior of the back propagation algorithm for adaptive estimation of the weights associated with perceptron based neural network models.

1. INTRODUCTION

In the last 40 years many artificial neural network models have been proposed [1-10]. All of them try to achieve high computation rates via massive parallelism. A hallmark of such models is significant in built redundancy. This makes them relatively robust to localized physical destruction of component units, and intrinsically fault tolerant. In recent years, some newly proposed models and the development of new analog VLSI implementation techniques, have infused fresh interest in the study of neural networks . Examples of their current applications include speech and image processing and pattern classification in general.

One of the earliest neural network models, derives from a perceptron structure [1], an example of which is given in figure 1. Basically, such models consist of several layers of node each node representing a component neuron. The output of each neuron is its state, taking values between 0 to 1 according to the formula (sigmoidal function)

$$\text{neural state} = \frac{1}{1+\exp(-s)} \tag{1}$$

Here s is the amount by which the total stimulus received by this neuron, exceeds a given threshold. If s is large the neural state is close to one. For small s the state is close to zero. A neuron in a given layer receives stimulus from neurons in the layer immediately below it. This stimulus in turn equals a weighted sum of the states of the neurons providing the excitation, the weights representing the strength of the relevant interconnections. These weights thus completely define the network, and their estimation is critical to understanding the network properties.

One difficulty with such an estimation is that only the outputs of the last layer and inputs to the first layer are observable. Both the inputs and the outputs of the intermediate layers are not directly accessible. The intermediate layers are thus referred to as hidden layers. An adaptive weight estimation technique, known as the back propagation model, has been studied in [2]. This technique adaptively updates the weight estimates by a steepest descent minimization of the instantaneous sum of the squares of the difference between desired and actual outputs of the last layer. Though this algorithm has been successfully implemented, very little is known of its convergence properties. In section 2 of this paper, we give a definitive analysis of the case where there are no hidden layers. Such a case is important in its own right as it can be used as a maximum likelihood Gaussian Classifier [1]. In section 3 we give a local analysis of the case where there is only one hidden layer. The analysis is preliminary, but reveals certain interesting error models which to our knowledge have not been encountered before. Section 4 is the conclusion.

2. CONVERGENCE WITH NO HIDDEN LAYERS

Consider the network of figure 1, with no hidden nodes. Denote the outputs of the n nodes in the last layer as $y_i(k)$, k denoting the k^{th} time instant. Denote $x_i(k)$, $i = 1,...,m$, as the input stimulus to the i^{th} node of the first layer. The $x_i(k)$ will be assumed to be binary. In the pattern classification problem, for example, they can be viewed together as an input pattern to be classified. Denoting $X(k) = [x_1(k),....,x_m(k)]^T$ and $Y(k) = [y_1(k),....,y_n(k)]^T$, we assume that the input output description satisfies

$$N(k) = \mathbf{W}\, X(k) + B \tag{2}$$

$$Y(k) = f[N(k)] \tag{3}$$

where $B = [b_1(k),....,b_n(k)]^T$, $\mathbf{W}$ is an n x m matrix with elements w_{ij} representing the strength between the j^{th} node in the input layer and the i^{th} node in the output layer, $N(k) = [n_1(k),....,n_n(k)]^T$ and f(s) for scalar s represents precisely the sigmoidal function of (1), and for vector arguments are applied in an element by element fashion. The $-b_i$'s represent the threshold associated with the i^{th} output node. The need for time indexing the signals arises from the requirement that to adapt the network weights successfully, one must vary the input patterns. One may view the set of patterns used for this purpose as training patterns.

We set up the network model as

$$\hat{N}(k) = \hat{\mathbf{W}}(k-1)\, X(k) + \hat{B}(k-1) \tag{4}$$

$$\hat{Y}(k) = f[\hat{N}(k)] \tag{5}$$

where $\hat{\mathbf{W}}(k-1)$ and $\hat{B}(k-1)$ are estimates of $\mathbf{W}$ and B at time (k-1) respectively. $\hat{Y}(k)$ is the output of adaptive net at time k. The objective is to adapt the $\hat{\mathbf{W}}(k-1)$ and $\hat{B}(k-1)$, according to the applied patterns X(k) and the difference between the desired and actual outputs. This is done by the steepest descent minimization of $\frac{1}{2}\,||\,[Y(k) - \hat{Y}(k)]||\,_2^2$. The resulting algorithm [1] is given as

$$\hat{w}_{ij}(k) = \hat{w}_{ij}(k-1) + \mu[y_i(k) - \hat{y}_i(k)]\ f'\ (\hat{n}_i(k))\ x_j(k) \tag{6}$$

$$\hat{b}_i(k) = \hat{b}_i(k-1) + \mu[y_i(k) - \hat{y}_i(k)]\ f'\ (\hat{n}_i(k)) \tag{7}$$

where f' (.) denotes the derivative with respect to the arguments, and μ is a positive adaptation gain. Notice, the similarity between (6) and (7) and the LMS adaptive estimator [11]. The convergence analysis too is similar to that for LMS [12], though the function f' (.) introduces certain complications. As in [12] we will argue that, the exponential convergence of (6) and (7) requires that the training patterns be sufficiently varied, or more precisely that the regressor vector $Z(k) = [\ X^T(k),1\ ]^T$, span the entire (m + 1) space, and that the adaptation gain μ be chosen smaller then a value to be specified presently.

Since $\hat{n}_i(k)$ depends only on $[\hat{w}_{i1}(k-1),....,\hat{w}_{im}(k-1),\hat{b}_i(k-1)]^T$, we can group the equations involving these weights together; and thus can write the adaptation equations as

$$\hat{\Theta}_i(k) = \hat{\Theta}_i(k-1) + \mu\ f'\ (\hat{n}_i(k))\ [y_i(k) - \hat{y}_i(k)]\ Z(k)$$

where $\hat{\Theta}_i(k) = [\hat{w}_{i1}(k),....,\hat{w}_{im}(k),\hat{b}_i(k)]^T$. If we define $\Theta_i = [w_{i1},....,w_{im},b_i]^T$, and $\tilde{\Theta}_i(k) = \Theta_i - \hat{\Theta}_i(k)$, then we can write

$$\tilde{\Theta}_i(k) = \tilde{\Theta}_i(k-1) - \mu\ \varsigma_i(k)\ Z(k) \tag{8}$$

where $\varsigma_i(k) = f'\ (\hat{n}_i(k))\ [y_i(k) - \hat{y}_i(k)]$, so we have,

$$||\,\tilde{\Theta}_i(k)||^2 = ||\,\tilde{\Theta}_i(k-1)||^2 - 2\,\mu\ \varsigma_i(k)\ \tilde{\Theta}_i(k-1)^T\ Z(k) + \mu^2\ \varsigma_i(k)^2\ ||\,Z(k)||^2$$

It follows that $\tilde{\Theta}_i(k-1)^T\ Z(k) = [n_i(k) - \hat{n}_i(k)]$, whence

$$||\,\tilde{\Theta}_i(k)||^2 = ||\,\tilde{\Theta}_i(k-1)||^2 - \mu\left[\frac{2\ [n_i(k) - \hat{n}_i(k)]}{\varsigma_i(k)\ ||\,Z(k)||^2} - \mu\right]\varsigma_i(k)^2\ ||\,Z(k)||^2 \tag{9}$$

For a binary input vector, we have $1 \leq ||\,Z(k)||^2 \leq (m + 1)$, for all k. Since $f(\hat{n}_i(k))$ is the sigmoidal function (1), we have $f'\ (\hat{n}_i(k)) = f(\hat{n}_i(k))\ (1 - f(\hat{n}_i(k)))$, and so $0 \leq f'\ (\hat{n}_i(k)) \leq 0.25$, for all k; as by definition $0 \leq f(\hat{n}_i(k)) \leq 1$, for all k. Using these facts and some manipulations, we can show that

$$\left[\frac{2\ [n_i(k) - \hat{n}_i(k)]}{\varsigma_i(k)\ ||\,Z(k)||^2}\right] \geq \frac{32}{(m+1)}$$

Thus choosing

$$0 < \mu < \frac{32}{(m+1)} \leq \left[\frac{2\ [n_i(k) - \hat{n}_i(k)]}{\varsigma_i(k)\ ||\,Z(k)||^2}\right] \tag{10}$$

we can guarantee due to (9) that, for all i

$$||\,\tilde{\Theta}_i(k)||\, \leq\, ||\,\tilde{\Theta}_i(k-1)|| \tag{11}$$

and

$$\lim_{k\to\infty} \varsigma_i(k) = 0$$

From the definition of $\varsigma_i(k)$, this implies that either

$$\lim_{k\to\infty} |\,[y_i(k) - \hat{y}_i(k)]| = 0 \tag{12}$$

or

$$\lim_{k\to\infty} f'\ (\hat{n}_i(k)) = 0 \tag{13}$$

The definition of the sigmoidal function implies that (13) can happen only due to unbounded $\hat{n}_i$, which of course is impossible due to (11) and the fact that X(k) is binary. Thus (12) holds and we have shown that the actual network outputs asymptotically approach their desired values. This of course only shows output convergence. For weight convergence we need the condition that there exist positive α_1,T, such that for all k

$$\sum_{i=k}^{k+T-1} Z(k)\ Z^T(k) \geq \alpha_1\ I \tag{14}$$

In that event one can prove that existence of a γ, $0 < \gamma < 1$ such that, for all k, and i $\in$ [1,...,m]

$$||\,\tilde{\Theta}_i(k+T)|\,|^2 \leq (1 - \gamma)\,||\,\tilde{\Theta}_i(k)|\,|^2 \tag{15}$$

Details of the proof are omitted. But clearly (15) implies that $||\,\tilde{\Theta}_i|\,|$ decays exponentially to zero.

3. NETWORKS WITH ONE HIDDEN LAYER

In studying a network with one hidden layer with the input, hidden and output layers having p, m, n nodes respectively, we will assume that the desired input output relationship is

$$N^h(k) = \mathbf{W}^h\ X(k) + B^h$$

$$H(k) = f[N^h(k)]$$

$$N(k) = \mathbf{W}\ H(k) + B$$

$$Y(k) = f[N(k)]$$

Here $X(k) = [x_1(k),....,x_p(k)]^T$, is the input vector at time k, $N^h(k) = [n_1^h(k),....,n_m^h(k)]^T$, $H(k) = [h_1(k),....,h_m(k)]^T$, is the output of the hidden layer, $N(k) = [n_1(k),....,n_n(k)]^T$, $Y(k) = [y_1(k),....,y_n(k)]^T$, is the desired output vector for input pattern X(k), $\mathbf{W}^h = [w_{ij}^h]$, is the m x p weight matrix for the input layer of desired net, $B^h = [b_1^h,....,b_m^h]^T$, is the threshold vector for the input layer of desired net, $\mathbf{W} = [w_{ij}]$, is the n x m weight matrix for the hidden layer of desired net, and $B = [b_1,....,b_n]^T$, is the threshold vector for the hidden layer of desired net.

The adaptive neural net can then be formulated as

$$\begin{aligned}\hat{N}^h(k) &= \hat{\mathbf{W}}^h(k-1)\, X(k) + \hat{B}^h(k-1) \\ \hat{H}(k) &= f[\hat{N}^h(k)] \\ \hat{N}(k) &= \hat{\mathbf{W}}(k-1)\, \hat{H}(k) + \hat{B}(k-1) \\ \hat{Y}(k) &= f[\hat{N}(k)]\end{aligned} \tag{16}$$

where $\hat{\mathbf{W}}^h(k-1)$, $\hat{B}^h(k-1)$, $\hat{\mathbf{W}}(k-1)$ and $\hat{B}(k-1)$ are estimates of $\mathbf{W}^h$, B^h, $\mathbf{W}$ and B respectively at time (k-1). Then with μ the adaptation gain and,

$$\hat{\delta}_i(k) = f'\ (\hat{n}_i(k))\ [y_i(k) - \hat{y}_i(k)] \tag{17}$$

the update equations become [1],

$$\hat{w}_{ij}(k) = \hat{w}_{ij}(k-1) + \mu\ \hat{\delta}_i(k)\hat{h}_j(k)$$

$$\hat{b}_i(k) = \hat{b}_i(k-1) + \mu\ \hat{\delta}_i(k)$$

$$\hat{w}_{ij}^h(k) = \hat{w}_{ij}^h(k-1) + \mu \sum_{l=1}^{n} \hat{\delta}_l(k)\ \hat{w}_{li}(k-1)\quad f'\ (\hat{n}_i^h(k))\ x_j(k)$$

$$\hat{b}_i^h(k) = \hat{b}_i^h(k-1) + \mu \sum_{l=1}^{n} \hat{\delta}_l(k)\ \hat{w}_{li}(k-1)\quad f'\ (\hat{n}_i^h(k)),$$

More concisely, these can be written as,

$$[\hat{\mathbf{W}}(k),\hat{B}(k)] = [\hat{\mathbf{W}}(k-1),\hat{B}(k-1)] + \mu\hat{\delta}(k)\ \hat{Z}_2^T(k)$$

$$[\hat{\mathbf{W}}^h(k),\hat{B}^h(k)] = [\hat{\mathbf{W}}^{(k-1)},\hat{B}^{(k-1)}] + \mu\ \hat{\mathbf{D}}(k)\ \hat{\mathbf{W}}^T(k-1)\ \hat{\delta}(k)\ Z_1^T(k)$$

where,

$$\hat{\delta}(k) = [\hat{\delta}_1(k),\ldots.,\hat{\delta}_n(k)]^T$$

$$\hat{Z}_2(k) = [\hat{h}_1(k),\ldots.,\hat{h}_m(k),1]^T$$

$$Z_1(k) = [x_1(k),\ldots.,x_p(k),1]^T$$

$$\hat{\mathbf{D}}(k) = \mathrm{diag}\ [f'\ (\hat{n}_1^h(k)),\ldots.,f'\ (\hat{n}_m^h(k))]$$

We now do a local analysis under the assumption that the initial difference between the actual and desired weights is "small". Define $\Delta\mathbf{W}(k) = \mathbf{W} - \hat{\mathbf{W}}(k)$, $\Delta\mathbf{W}^h(k) = \mathbf{W}^h - \hat{\mathbf{W}}^h(k)$, $\Delta B(k) = B - \hat{B}(k)$ and $\Delta B^h(k) = B^h - \hat{B}^h(k)$. Then neglecting high order terms in these quantities, we find that the update equations reduce to,

$$\begin{bmatrix}\Delta\mathbf{W}^T(k)\\ \Delta B^T(k)\end{bmatrix} = \begin{bmatrix}\Delta\mathbf{W}^T(k-1)\\ \Delta B^T(k-1)\end{bmatrix} - \mu\ Z_2(k)\ Z_1^T(k) \begin{bmatrix}\Delta\mathbf{W}^{h^T}(k-1)\\ \Delta B^{h^T}(k-1)\end{bmatrix} \mathbf{D}(k)\ \mathbf{W}^T\ \mathbf{D}_1^2(k)$$

$$- \mu\ Z_2(k)\ Z_2^T(k) \begin{bmatrix}\Delta\mathbf{W}^T(k-1)\\ \Delta B^T(k-1)\end{bmatrix} \mathbf{D}_1^2(k)$$

$$\begin{bmatrix} \Delta \mathrm{W}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}) \\ \Delta \mathrm{B}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}) \end{bmatrix} = \begin{bmatrix} \Delta \mathrm{W}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}-1) \\ \Delta \mathrm{B}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}-1) \end{bmatrix} - \mu\, \mathrm{Z}_1(\mathrm{k})\, \mathrm{Z}_1^{\mathrm{T}}(\mathrm{k}) \begin{bmatrix} \Delta \mathrm{W}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}-1) \\ \Delta \mathrm{B}^{\mathrm{h}^{\mathrm{T}}}(\mathrm{k}-1) \end{bmatrix} \mathbf{D}(\mathrm{k})\, \mathbf{W}^{\mathrm{T}}\, \mathbf{D}_1^2(\mathrm{k})\, \mathbf{W}\, \mathbf{D}(\mathrm{k})$$

$$- \mu\, \mathrm{Z}_2(\mathrm{k})\, \mathrm{Z}_1^{\mathrm{T}}(\mathrm{k}) \begin{bmatrix} \Delta \mathrm{W}^{\mathrm{T}}(\mathrm{k}-1) \\ \Delta \mathrm{B}^{\mathrm{T}}(\mathrm{k}-1) \end{bmatrix} \mathbf{D}_1^2(\mathrm{k})\, \mathbf{W}\, \mathbf{D}(\mathrm{k})$$

where

$$\mathrm{Z}_2(\mathrm{k}) = [\mathrm{h}_1(\mathrm{k}),\ldots,\mathrm{h}_{\mathrm{m}}(\mathrm{k}),1]^{\mathrm{T}}$$

$$\mathbf{D}(\mathrm{k}) = \mathrm{diag}\,[\mathrm{f}'\,(\mathrm{n}_1^{\mathrm{h}}(\mathrm{k})),\ldots,\mathrm{f}'\,(\mathrm{n}_{\mathrm{m}}^{\mathrm{h}}(\mathrm{k}))]$$

$$\text{and } \mathbf{D}_1(\mathrm{k}) = \mathrm{diag}\,[\mathrm{f}'\,(\mathrm{n}_1(\mathrm{k})),\ldots,\mathrm{f}'\,(\mathrm{n}_{\mathrm{n}}(\mathrm{k}))]$$

Notice that the above equations resemble, closely a set of coupled, time varying Lyapunov equations. To our knowledge such equations have not been encountered before particularly as some of the coefficient matrices are rank - 1.

By writing, columns of these error matrices into a vector $\tilde{\Theta}$, after some manipulations we can write the above equations in the following form,

$$\tilde{\Theta}(\mathrm{k}) = \tilde{\Theta}(\mathrm{k}-1) - \mu\, \mathrm{R}(\mathrm{k})\, \tilde{\Theta}(\mathrm{k}-1) \tag{18}$$

Here,

$$\tilde{\Theta}(\mathrm{k}) = [\Delta \mathrm{w}_{11}(\mathrm{k}),\ldots,\Delta \mathrm{w}_{1\mathrm{m}}(\mathrm{k}),\Delta \mathrm{b}_1(\mathrm{k}),\ldots,\Delta \mathrm{w}_{\mathrm{nm}}(\mathrm{k}),\Delta \mathrm{b}_{\mathrm{n}}(\mathrm{k}),\Delta \mathrm{w}_{11}^{\mathrm{h}}(\mathrm{k}),\ldots,$$

$$\Delta \mathrm{w}_{1\mathrm{p}}^{\mathrm{h}}(\mathrm{k}),\Delta \mathrm{b}_1^{\mathrm{h}}(\mathrm{k}),\ldots,\Delta \mathrm{w}_{\mathrm{mp}}^{\mathrm{h}}(\mathrm{k}),\Delta \mathrm{b}_{\mathrm{m}}^{\mathrm{h}}(\mathrm{k})]^{\mathrm{T}}$$

$$\mathrm{R}(\mathrm{k}) = \begin{bmatrix} \mathbf{A}_2(\mathrm{k})\, \mathbf{A}_5(\mathrm{k}) & \mathbf{A}_2(\mathrm{k})\, \mathbf{A}_3\, \mathbf{A}_4^{\mathrm{T}}(\mathrm{k})\, \mathbf{A}_1(\mathrm{k}) \\ \mathbf{A}_1(\mathrm{k})\, \mathbf{A}_4(\mathrm{k})\, \mathbf{A}_3^{\mathrm{T}}\, \mathbf{A}_2(\mathrm{k}) & \mathbf{A}_1(\mathrm{k})\, \mathbf{A}_6(\mathrm{k})\, \mathbf{A}_7^{\mathrm{T}}\, \mathbf{A}_8(\mathrm{k})\, \mathbf{A}_7\, \mathbf{A}_1(\mathrm{k}) \end{bmatrix}$$

and $\mathbf{A}_1(\mathrm{k}),\mathbf{A}_2(\mathrm{k}),\mathbf{A}_4(\mathrm{k}),\mathbf{A}_5(\mathrm{k}),\mathbf{A}_6(\mathrm{k})$ and $\mathbf{A}_8(\mathrm{k})$ are block diagonal matrices given by,

$$\mathbf{A}_1(\mathrm{k}) = \mathbf{I}_{\mathrm{p}+1} \otimes \mathbf{D}(\mathrm{k})$$

$$\mathbf{A}_2(\mathrm{k}) = \mathrm{diag}\,[\mathrm{f}'\,^2(\mathrm{n}_1(\mathrm{k})),\ldots,\mathrm{f}'\,^2(\mathrm{n}_{\mathrm{n}}(\mathrm{k}))] \otimes \mathbf{I}_{\mathrm{m}+1}$$

$$\mathbf{A}_3 = \mathbf{W} \otimes \mathbf{I}_{\mathrm{m}+1}$$

$$\mathbf{A}_4(\mathrm{k}) = \mathbf{I}_{\mathrm{m}} \otimes [\mathrm{Z}_1(\mathrm{k})\mathrm{Z}_2^{\mathrm{T}}(\mathrm{k})]$$

$$\mathbf{A}_5(\mathrm{k}) = \mathbf{I}_{\mathrm{n}} \otimes [\mathrm{Z}_2(\mathrm{k})\mathrm{Z}_2^{\mathrm{T}}(\mathrm{k})]$$

$$\mathbf{A}_6(\mathrm{k}) = \mathbf{I}_{\mathrm{m}} \otimes [\mathrm{Z}_1(\mathrm{k})\mathrm{Z}_1^{\mathrm{T}}(\mathrm{k})]$$

$$\mathbf{A}_7 = \mathbf{W} \otimes \mathbf{I}_{\mathrm{p}+1}$$

$$\mathbf{A}_8(\mathrm{k}) = \mathrm{diag}\,[\mathrm{f}'\,^2(\mathrm{n}_1(\mathrm{k})),\ldots,\mathrm{f}'\,^2(\mathrm{n}_{\mathrm{n}}(\mathrm{k}))] \otimes \mathbf{I}_{\mathrm{p}+1}$$

Then some standard analysis, see for example, [13], shows that for small enough μ, $\tilde{\Theta}(\mathrm{k})$ in (18) is exponentially convergent provided there exists T and positive α_1 such that for all k

$$\sum_{\mathrm{i}=1}^{\mathrm{T}-1} [\mathbf{R}(\mathrm{k}+\mathrm{i}) + \mathbf{R}^{\mathrm{T}}(\mathrm{k}+\mathrm{i})] \geq \alpha_1\, \mathrm{I} \tag{19}$$

Basically, necessary for the satisfaction of (19), is the requirement, that the weights be locally, uniquely identifiable. This in turn will require that the input vector X(k) be sufficiently varied. However notice, **R**, depends on the desired value of the weight estimates in the last layer. Thus certain structural conditions may preclude (19) even if X(k) is persistently spanning. As an example consider the structure in figure 2, with only one node per layer, i.e. n = m = p = 1. Then **R**(k) reduces to

$$f'^{\,2}(n_1(k))\, \Gamma(k)\, \Gamma^T(k)$$

where

$$\Gamma(k) = [wf'\ (n_1^h(k))x_1(k),\ wf'\ (n_1^h(k)),\ h_1(k),\ 1]^T$$

Then with binary x, the relationships between h, n^h and x, force Γ(k), not to span the entire space $\mathbf{R}^4$. Thus as long as x(k) is restricted to be binary, (19) cannot be satisfied. Notice also that in this case with binary inputs, and all four quantities (two weights and two thresholds), unknown the network can independently be shown as nonidentifiable. With non-binary x(k) however, sequences of x(k) can be formed to make (19) hold. These sequences will also result in the network being identifiable.

4. CONCLUSION

We have considered the convergence behavior of perceptron based neural networks having one and zero hidden layers.

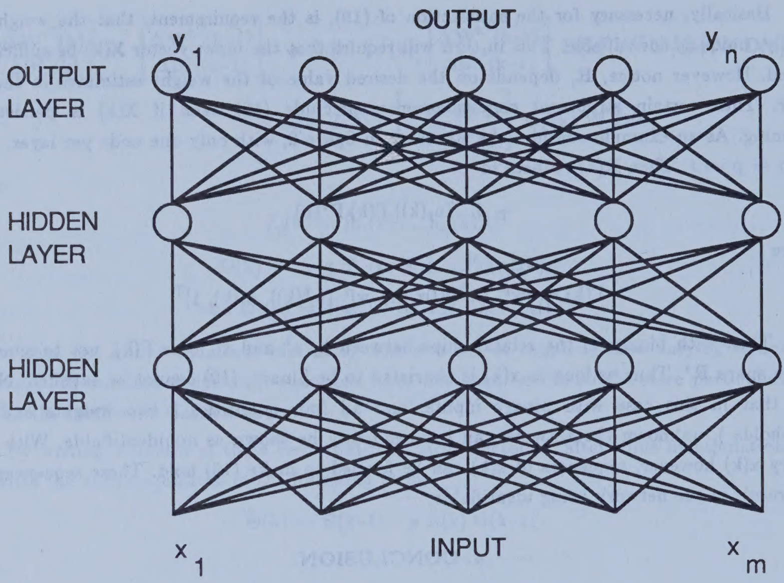

OUTPUT
y_1
y_n
OUTPUT LAYER
HIDDEN LAYER
HIDDEN LAYER
x_1
INPUT
x_m

FIGURE 1. A MULTILAYER PERCEPTRON

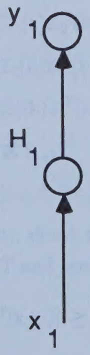


FIGURE 2 EXAMPLE WITH ONE HIDDEN LAYER AND ONLY ONE NODE PER LAYER

REFERENCES

[1] R.P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine,* April 1987, pp 4-22.

[2] R.J. Williams, "Adaptive Connectionist System", *Proceedings of the fifth Yale Workshop on Applications of Adaptive Systems Theory,* May, 1987, pp 102-107

[3] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation", in D.E. Rumelhart & J.L. McLelland, (Eds), *Parallel Distributed Processing : Explorations in the Microstructure of Cognition,* MIT Press (1986)

[4] G.A. Carpenter and S. Grossberg, "Neural Dynamics of Category Learning and Recognition : Attention, Memory Consolidation, and Amnesia", in J. Davis, R. Newburgh, and E. Wegman, (Eds), *Brain Structure, Learning and Memory,* AAAS Symposium Series, 1986

[5] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA,* vol. 79, pp 2554-2558, April 1982.

[6] M. Minsky and S. Papert, *Perceptrons : An Introduction to Computational Geometry,* MIT Press (1969).

[7] E. Kandel and J.H. Schwartz, *Principles of Neural Science,* Elsevier, New York 1985.

[8] R.P. Lippmann, B. Gold and M.L. Malpass, "A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification", *MIT Lincoln Laboratory Technical Report,* TR-769.

[9] J.J. Hopfield and D.W. Tank, "Computing with Neural Circuits : a model", *Science,* pp 625-633, August 1986.

[10] M.A. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks", *IEEE Trans. Syst. Man and Cyb.,* SMC-13, 815-826, 1983.

[11] B. Widrow and M.E. Hoff Jr, "Adaptive Switching Circuits", *IRE, WESCON Conv. Rec.,* pt. 4, pp 96-104, 1960

[12] A. Weiss and D. Mitra, "Digital Adaptive Filters : Conditions for Convergence, Rates of Convergence, Effects of noise and Errors arising from Implementation", *IEEE Trans. on Info. Thy.,* vol IT-25, pp 637-652, Nov. 1979.

[13] W.A. Sethares, I.M.Y. Marcels, B.D.O. Anderson, C.R. Johnson Jr and R.B. Bitmead, "Excitation Conditions for Signed Regressor LMS Adaptation", To appear *IEEE Transactions on Circuits and Systems.*

ON THE DESIGN OF NEURAL NETS

Fathi M.A. Salam, M. R. Choi, Y. Wang, and B. Javidi

Department of Electrical Engineering and Systems Science,
Michigan State University,
East Lansing, MI 48824

Summary

There are three main limiting characteristics in using digital or sequential computers in certain applications. Firstly, the computational speed of the available microprocessors is limited by their synchronous operations. Secondly, the use of logical digital devices as building blocks results in the loss of accuracy on the one hand, and in the requirement of a large chip-area on the other. Thirdly, the prevalent design of associative memory (content-addressable memory or CAM) requires the precise knowledge of the address being retrieved. (The content of CAM is usually an address of an entire memory in the RAM.)

To circumvent these limitations, the following goals can be identified. One goal is to employ asynchronous parallel processing in order to achieve faster computational speeds. A second goal is to implement analog circuits comprised of resistive elements and operational amplifiers as basic building blocks; consequently, reducing the required chip-area. And a third goal is to design an associative memory (CAM) which would correctly yield any of its contents from mere partial information.

Neural networks, as found in the nervous system, possess immense computational powers which is attributed largely to their (feedback) interconnectivity properties. We study the architecture of neural networks in order to design and *systematically* specify the functional organization of an associative array processor. Consequently, a systematic and accurate design of associative memory can be realized. Our design transforms into a systematic procedure which correctly programs the interconnection among neurons so that it corresponds to any set of prespecified memories precisely. The design also guarantees retrievability of the (prespecified) stored memories. This paper summarizes sum of our results. Then a new implemetation using real-time optical computing is presented at the end.

1. INTRODUCTION

There are various application tasks, often called random problems, where conventional digital computers can not compete with the performance of humans. We mention such two tasks which are of interest to us here. The first is pattern recognition, e.g., identifying a familiar face in a croud; the second is the process of associative memory, e.g., being able to quickly "remember" one's face or name when only partial information or some distinguishing characteristics are given.

The limitation of conventional digital computers and microprocessors in these applications stems from their architectural designs. We identify three main limitations. First, the computational speed of the available microprocessors is limited by their synchronous operations. Second, the use of logical digital devices as building blocks results in the loss of accuracy on the one hand, and in the requirement of a large chip-area on the other. Thus, restricting the number of building blocks in each microprocessor or computer. Third, the prevalent design of associative memory (content-addressable memory or CAM) requires the precise knowledge of the information being searched. In addition, the hardware composing any CAM is usually quite expensive.

Parallel processing offers a solution to the first limitation and it has been utilized successfully in, among other applications, vector-vector and vector-matrix multiplications. It does not offer any solutions to the other limitations, however. In fact, it is recognized that no matter how fast conventional computers would become, it is unlikely that they could outperform the human performance in tasks such as pattern recognition or associative memory. It is only logical therefore that for such tasks the architecture of the microprocessor or the computer should emulate that of the brain. Many workers have proposed electrical circuits that model the highly interconnected nerves system and the brain. These circuits are often referred to as *neural nets*. One such circuit has been proposed by Hopfield and it is believed to be promising in circumventing all limitations of conventional computers. Our emphasis will be primarily on Hopfield's neural network.

The architecture or topology of neural nets has basically been established via the work of Hopfield and his collaboraters, as well as other researchers. The formulation of Hopfield basically translates the operation of a neural network into a minimization of a scalar function or, equivalently, into a gradient dynamic system. It implicitly relates the desired memories which would be stored into the network to the values of interconnecting transconductances. Some procedures that characterize this relationship have been proposed by Hopfield; they are based on intuitive arguments and physical reasoning, however. Although they have worked well in some case studies, they have been found to fail in other cases as well. As of yet, a comprehensive understanding of the neural network model, proposed by Hopfield as well as all other models [8], is lacking. It is essential, therefore, that a theoretical framework for neural nets be established. In addition, one has to accomodate the prevalent technologies and pursue a balance of theory and practice. It is hoped that such an approach would lead to solving the real challenge in neural nets today, i.e., how to *choose* or *"program"* the interconnections so as to render a relative configuration of the neural network which corresponds to a specified set of stored memories .

2. BACKGROUND ON NEURAL NETS

We have carefully studied the neural network proposed by Hopfield and his coworkers in the hope that the principles of its operation can be successfully applied in designing machines capable of solving random problems. The random problems whose solution requires knowledge of essentially every possible state of a system can not be precisely defined and they are difficult to solve using conventional computers. Devices designed to model the working of neural network would basically consist of a large number of simple processors (or neurons) that are highly interconnected and work asynchronously and parallely.

In a single neuron, a train of action potentials is propagating pulses of electro- chemical activity. If the neuron has a strong input, these pulses are generated at a high rate. If the input is weak or absent, they are generated at very low rate. The mean rate at which these action potentials are generated results in a smooth function of the mean membrane potential, say $g(u_i)$. Since the synapses are activated by arriving action potentials, $V_i = g(u_i)$ becomes an input and output relationship for a neuron, where u_i could be thought of as the mean soma potential of a neuron from the total effect of its excitatory and inhibitory inputs, and V_i could be viewed as the short-term average of the firing rate of the cell *i*.

We have employed Hopfield's model as our basic architecture to reach our goals. In this model, neurons are processing devices and each neuron has two states: 0 means that the neuron is not firing and 1 means that it is firing at maximum rate. The connection T_{ij} is defined by synaptic weight which describes the extent to which the input of neuron *i* is driven by the output from neuron *j*. If the synapse is excitatory, the output from neuron *j* will drive neuron *i* to produce more output, i.e. T_{ij} is positive. If it is inhibitory, less output will be produced and T_{ij} will be negative. A normal, respectively an inverting, operational amplifier is implemented for $T_{ij}>0$, respectively $T_{ij}<0$ respectively.

In a biological system, u_i will lag behind the instantaneous outputs V_j of the other cells because of the input capacitance C_i of the cell membrane, the transmembrane resistance R_i, and the finite conductance T_{ij}. Thus the rate of change of u_i is determined by the following resistance-capacitance charging equation.

$$C_i\left[du_i/dt\right] = \sum_j T_{ij}V_j - u_i/R_i + I_i \tag{0.1}$$

$$u_i = g_i^{-1}\left[V_i\right]$$

where $R_{ij} = \left|\frac{1}{T_{ij}}\right|$, $R_i^{-1} = \rho_i^{-1} + \sum_j R_{ij}^{-1}$, which is the net input impedance and C_i is the total input capacitance of the amplifier i.

We can now construct an electrical network composed of many (operational) amplifiers that are connected via resistive elements, see [1-5].

A recent article [8] surveys the design of classifiers using neural nets, particularly, the Hopfield net and the Hamming net. These two nets have strong resemblence to one another at least conceptually. These nets have been tested on problems where inputs are generated by selecting an exemplar whose pattern is most representative of each class and reversing its bit values randomly and independently. Although the nets were given corrupted (or partial) information as their inputs, they continued to process until eventually their outputs became the examplar pattern that is closest to the unknown input.

Another recent article [7] has argued for the possibility of designing optical neural computers. Optical technology appears to be natural for the neural architecture, particularly its suitability for handling the dense interconnectivity of neural nets. An alternative to the implementation of optical technology, there are efforts to employ semiconductor technology. For instance, a design methodology as well as several configurations of CMOS operational amplifiers are the subject of the recent book [12].

3. THE MATHEMATICAL FORMULATION OF NEURAL NETS

Here we summarize some results [17]. Due to space limitation, we tend to be informal in our presentation, emphasizing the main ideas only.

3.1. Conceptual Framework

We list our conceptual framework of neural nets in the following summary.

1. All solutions must converge to a compact set.
2. The limit sets consist of only equilibrium points.
3. There must be a simple relationship between the interconnects, T_{ij}'s, and the corresponding (stored) memories.
4. The overall dynamic system must be structurally stable. So in implementation, changes in parameter values do not change the properties of the system.

3.2. Properties of the symmetric matrix T

We now present some properties of the symmetric matrix T which are essential in the design process of neural networks.

Let T be an nxn symmetric matrix. Let its eigenvalues $\lambda_1, \ldots, \lambda_n$ be distinct, and let $e_1, \ldots, e_n$ be a corresponding set of eigenvectors. Finally, let $u_i = \frac{1}{\|e_i\|}e_i$, $i = 1, \ldots, n$ be the normalized eigenvectors. Then the following statements are true (see [16] for example.):

1. $\lambda_1, \ldots, \lambda_n$ are real.
2. T has a diagonal Jordan-form representation.

3. $e_1, \dots, e_n$ are orthogonal.
4. $u_1, \dots, u_n$ are orthonormal.
5. The matrix $P \equiv [u_1, \dots, u_n]$ is orthogonal, i.e.

$$P P^T = P^T P = I$$

6. $T = P \Lambda P^T$, where $\Lambda = \text{diag}[\lambda_1, \dots, \lambda_n]$.

3.3. Gradient Systems

The mathematical model for a neuronetwork is

$$C_i \frac{du_i}{dt} = \sum_j T_{ij} v_j + I_i - \frac{u_i}{R_i} \tag{1.1i}$$

$$v_i = g_i(u_i). \tag{1.1ii}$$

In compact matrix form, (1.1) can be written as

$$C \frac{d}{dt} u = Tv + I - R^{-1} u , \tag{1.2i}$$

$$v = G(u) = [g_1(u_1), \dots, g_n(u_n)]^T \tag{1.2ii}$$

The so-called potential energy function, which is a first integral of (1.2), can be derived as

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij} v_i v_j - \sum_i I_i v_i + R_i^{-1} \sum_i \int_0^{v_i} g^{-1}(\xi_i)\, d\xi_i , \tag{1.3i}$$

or in compact vector form

$$E = -\frac{1}{2} V^T T V - I^T V + [R^{-1}]^T \hat{F}(v) , \tag{1.3ii}$$

where $\hat{F}(v) = [\hat{f}_1(v_N), \dots, \hat{f}_N(v_N)]^T$ and $\hat{f}_i(v_i) = \int_0^{v_i} g^{-1}(\xi_i)\, d\xi_i$.

Hence the dynamic equations of the neural network model can be written as

$$C \frac{d}{dt} u = -\nabla E(v) \tag{1.4i}$$

$$v = G(u) \tag{1.4ii}$$

or

$$C(v) \dot{v} \triangleq (C \frac{d}{dv} G^{-1})\, \dot{v} = -\nabla E(v) \tag{1.5}$$

Eq (1.5) describes a gradient-like system.

We now present some basic properties of system (1.5), see also [16].

Theorem 1 : $\dot{E}(v) \le 0$ for any $v \in V$; and $\dot{E}(v) = 0$ if and only if v is an equilibrium of (1.5).

proof : Using the chain rule

$$\begin{aligned} \dot{E}(v) &= D_1 E(v)\, \dot{v} \\ &= < \nabla E(v) , - C(v)^{-1} \nabla E(v) > \\ &= - C(v)^{-1} \,|\, \nabla E(v) |^2 \le 0 \end{aligned}$$

Thus proving the theorem.

Corollary 1 : Let v^i be an isolated local minimum of E, then v^i is a (locally) asymptotically stable equilibrium of the gradient system (1.5).

proof : Consider the function

$W(v) = E(v) - E(v^i)$.

Then

$W(v^i) = 0$,

and for all v in the set $S_1 = \{ v \,|\, \| v - v^i \| < \delta \text{ for } 0 < \delta \ll 1 \}$, one gets

$$W(v) = W(v^i) + < D_1 W(v^i) \, , \, (v - v^i) > + (v - v^i)^T D_2 W(v^i) \, (v - v^i)$$

$$= < \nabla E(v^i) \, , \, (v - v^i) > + (v - v^i)^T D_2 W(v^i) \, (v - v^i)$$

$$= < (v - v^i) \, , \, D_2 W(v^i) \, (v - v^i) >$$

$$> 0 \qquad \text{iff} \; \| v - v^i \| > 0$$

Therefore, $W(v)$ is a candidate Liapunov function which is locally positive definite. Noting that along trajectories of (1.5) near v^i

$\dot{W}(v) = \dot{E}(v) = - C(v)^{-1} \, | \nabla E(v) |^2 \leq 0$,

establishes the corollary.

Theorem 2 : At every equilibrium of (1.5), the eigenvalues of the linearization of (1.5) are real.
Proof : The linearization of the left hand side of (1.5) at an equilibrium v^i gives the matrix

$$E := - \left[\frac{\partial^2 E}{\partial v_i \partial v_j} (v^i) \right] ,$$

which is symmetric.

We now show that all trajectories in the U^n-space converge to a bounded region. Correspondingly all trojectories in the V^n- space converge to a bounded region within the hypercube $[-\Delta \, , \, \Delta]^n$.

Consider the globally positive definite function W, where M is positive definite $(M \hat{=} R\, C(v))$

$$W(u) = \frac{1}{2} u^T R \, C(v) \, u$$

Then along trajectories, we have

$$\dot{W}(u) = u^T R \, C(v) \, [C^{-1}(v) \, (-R^{-1} u + Tv + I)]$$

$$= - u^T u + u^T R \, (Tv + I)$$

$$\leq - u^T u + | u^T R \, L | \; \leq \; - \| u \|^2 + \| u \| \, \| R L \| \; = \; - \| u \| \, [\, \| u \| - \| R L \| \,] ,$$

$\left(\text{where } L = [L_1 \, , \ldots , \, L_N]^T \text{ and } L_i \equiv \Delta_i \sum_{ij} | T_{ij} | + I_i . \right)$

Therefore,

$\dot{W}(u) < 0$,

for u outside of the sphere defined by

$$\rho = \left\{ u \in U^n \,|\; \quad \| u \|_2 \leq \| R \, L \|_2 \right\} .$$

Consequently all solutions must converge to the set ρ.

3.4. The Critical Points or The Equilibria

Observe that when the bias, $I = 0$, the vector 0 is always an equlibrium: simply plug 0 into equation (1.5) to verify that. In general let $v^1, \ldots, v^m$ be nonzero desired equilibria. Then, for each $i = 1, \ldots, m$, they must satisfy (see (1.2)):

$$0 = Tv^i - F(v^i) \tag{1.6}$$

where the bias current has been set to zero.
Eq (1.6), leads to

$$T[v^1, \ldots, v^m] = [F(v^1), \ldots, F(v^m)] \tag{1.7}$$

Then (1.7) leads to

$$T = [F(v^1), \ldots, F(v^m)][v^1, \ldots, v^m]^T \left\{ [v^1, \ldots, v^m][v^1, \ldots, v^m]^T \right\}^{-1} \tag{1.8}$$

Since we desire T to be symmetric, we set $T = M M^T$, where $M > 0$. Then

$$T = MM^T$$

$$= [F(v^1) \cdots F(v^m)][v^1 \cdots v^m]^T \left\{ [v^1 \cdots v^m][v^1 \cdots v^m]^T \right\}^{-1}$$

$$= \left\{ [v^1 \cdots v^m][v^1 \cdots v^m]^T \right\}^{-1} [v^1 \cdots v^m][F(v^1) \cdots F(v^m)]^T \tag{1.9}$$

which, after rearrangement, leads to the condition

$$[v^1 \cdots v^m][v^1 \cdots v^m]^T [F(v^1) \cdots F(v^m)][v^1 \cdots v^m]^T$$
$$= [v^1 \cdots v^m][F(v^1) \cdots F(v^m)]^T [v^1 \cdots v^m][v^1 \cdots v^m]^T$$

The above equality is satisfied if

$$[v^1 \cdots v^m]^T [F(v^1) \cdots F(v^m)] = [F(v^1) \cdots F(v^m)]^T [v^1 \cdots v^m]$$

or

$$F^T(v^i)v^j = F^T(v^j)v^i, \quad i \neq j, \tag{1.10}$$

which is our general condition on the stored memories, given the nonlinear vector function $F(\cdot)$.

The equilibria satisfy

$$Tv = F(v) = [f_1(v_1) \cdots f_n(v_n)]^T \tag{1.11}$$

$$Tv = \begin{bmatrix} \bar{f}_1(v_1) & . & . & 0 \\ \vdots & & \ddots & \vdots \\ 0 & . & . & . \\ 0 & . & . & \bar{f}_n(v_n) \end{bmatrix} v = \bar{F}(v)\, v$$

where $\bar{f}_i(v_i) \triangleq f(v_i)/v_i$ is an even function of v_i.
or

$$A(v)v \triangleq [T - \bar{F}(v)]\, v = 0 \tag{1.12}$$

Lemma 1: $A(v)$ is an even function of v.
proof : $A(v) = [T - \bar{F}(v)]$, where $\bar{F}(v)$ is an even function of v. Thus $A(-v) = [T - \bar{F}(-v)] = [T - \bar{F}(v)] = A(v)$.

The solutions of (1.12) determine the equilibria. The vector $v = 0$ is an equilibrium since it satisfies (1.12). It is isolated since

$$[\, T - D_1 F(v)\,]\big|_{v=0}$$

nonsingular by assumption.

Now suppose v^* is a nonzero equilibrium, then we have

$$A(v^*)v^* = [\, T - \bar{F}(v^*)\,]\, v^* = 0$$

Since $v^* \neq 0$, then the constant matrix $A(v^*)$ has to be singular with (at least) a one dimensional null space defined by v^*.

Theorem 3 : If v^* is a (nonzero) equilibrium of system (1.5), then so is $-v^*$.

proof : Continuing the above arguments, we proceed as follows. $A(-v^*)\,(-v^*) = -A(v^*)v^* = 0$.

3.5. Stability of The Equilibrium Point at 0

Since our system is a gradient system, then every trajectory must either converge to an equilibrium point or diverge. Since also the space $V = [-\Delta, \Delta]^n$, the diverging trajectories go to the boundry of the hybercube V. We choose the eigenvalues of T to be nonzero, ordered (real) scalars such that $\lambda_1 > \lambda_2 > \cdots > \lambda_m > 0 > \lambda_{m+1} > \cdots > \lambda_n$.

The zero vector is an equilibrium point since $\nabla E(0) = 0$. Its stablity is determined by the linearization of equation (1.5). We now derive the linearization. Write (1.5) as

$$\dot{v} = C^{-1}(v)\, [Tv + I - F(v)\,]$$

The linearization about the equilibrium v^i yields

$$\dot{\bar{v}} + h.o.t. = [\, C^{-1}(v^i) + C^{-1\prime}(v^i)\, \bar{v} + \cdots\,]\, [\, [\, Tv^i + I - F(v^i)\,] + [\, T\bar{v} - F(v^i)\bar{v}\,] + \cdots\,]$$

At $v^i = 0$, one obtains (after setting $I = 0$.)

$$\dot{\bar{v}} = [\, C^{-1}(v^i)T\,]\, \bar{v}$$

The equilibrium point 0 is required to be unstable. Consequently, it suffies if at least one of the eigenvalues of $[\, C^{-1}(v^i)T\,]$ is positive.

If we assume that the input-output relations for all $i = 1, \ldots, n$ are identical, then

$$C(v) = c(v)I,$$

where the scalar function $c(v) > 0$ for any v. Hence the eigenvalues of

$$[\, C^{-1}(v^i)T\,],$$

now become

$$\bar{\lambda}_i = \lambda_i\, c^{-1}(v^i).$$

Thus $\bar{\lambda}_i$ has the same sign as λ_i, where λ_i is an eigenvalue of T. Since $T = P\,\Lambda\, P^T$, then

$$[\, C^{-1}(v^i)T\,] = P\, [\, c^{-1}(v^i)\,\Lambda\,]\, P^T \equiv P\bar{\Lambda}P^T$$

$$= \sum_{i=1}^{n} \bar{\lambda}_i\, e_i e_i^T = \sum_{i=1}^{n} \lambda_i\, c^{-1}(v^i)\, e_i e_i^T .$$

Thus the eigenvalues of the matrix T determine the stability of the point 0.

4. IMPLEMENTATION VIA ELECTRO-OPTICS

We now present an optical implementation of the neural architecture specified by a modification of eqn. (1.1), i.e., the Hopfield model. The modification essentially amounts to a fixed translation so that the state space of the output vector v is $[\delta, \delta + 2\Delta]$, where $\delta \gg 0$.

To our knowledge, this is the first optical implementation of the continuous-time dynamic model of eqn. (1.1). In order to realize the implementation, we had to take certain practical issues into consideration. Primarily that: (1) all signals must be positive quantities, since signals in optics is carried out by the intensity of light; (2) although vector-matrix multiplications are performed in parallel, the result is given in terms of the square of the absolute values of the components of the resulting vector.

The real-time optical system is shown in Fig.1. Plane P_{11} contains an optically addressed spatial light modulator (*SLM*) denoted by SLM_1. The function of SLM_1 is to provide a nonlinear relationship between the input and output of the device.

The input and output of SLM_1 are v_i and u_i, respectively. Here, it is assumed that both u_i and v_i are positive quantities. The device used to provide the nonlinear relationship between u_i and v_i is a hybrid field-effect liquid crystal light valve, see [14]. The liquid crystal is operated in a hybrid field-effect mode which utilizes the twisted nematic effect to create a dark off-state and optical birefringence to create the high on-state. The sensitometry curve demonstrates the nonlinear behavior of the device is shown in [14]. The contrast ratio can be adjusted by using a number of different operational conditions. A contrast ratio of 100 : 1 can be achieved under typical conditions with the light valve approaching saturation with less than 100 $\mu w/cm^2$ of input light power. Presently, devices with limiting resolution of 40 lp/mm and aperture diameter of two inches are available.

The output signals v_i are imaged onto the image transducer SLM_2 located at plane P_{12}. SLM_2 is an electrically addressed spatial light modulator that contains the elements of the matrix T_{ij}. A programmable optical mask such as SLM_2 is necessary to provide flexibility in updating the elements of the matrix T_{ij}.

Commercially available liquid crystal televisions can be used to insert the values of T_{ij} in the system. The function of this device is similar to a liquid crystal light valve except that an LCTV is addressed electronically and LCLV is addressed optically. The values of T_{ij} can be calculated by an electronic computer interfaced with the LCTV. The matrix elements T_{ij}'s can be displayed on the LCTV and read-out coherently by a linearly polarized light. The commercially available LCTV's have space bandwidth products of larger than 128×128.

A vector matrix product operation is carried out between V and T, and the result is displayed at plane P_{13}. The charge coupled device CCD, located at plane P_{13}, detects the intensity of the output vector elements denoted by $|\sum v_i T_{ij}|^2$. Since the CCD detects the intensity of $\sum v_i T_{ij}$, a square root operation is performed electronically to obtain $\sum v_i T_{ij}$. The result is integrated in the time domain through the interface electronics to yield $\int \sum v_i T_{ij} dt$. The polarity of eachelement $\sum v_i T_{ij}$ is determined optically through the system L. The beam splitter BS_2 is used to apply elements of the vector to the system L which determines the direction of polarization for each element of the vector $\sum v_i T_{ij}$.

The integral of $\sum v_i T_{ij}$ is displayed on SLM_3 located at plane P_{21}. The sign of each element $\sum v_i T_{ij}$, determined by system L, is taken into account to provide accurate presentation of each value. SLM_3 is also an LCTV of the type described earlier. The integral function is read-out optically by linearly polarized light and is applied to the input of SLM_1, through the mirror M_1, and the beam splitter BS_3.

An optical feedback loop is constructed at the input of SLM_1 through the beam splitter BS_1. The vector u is imaged onto CCD_2 located at plane P_{22} to yield the intensity of u denoted by $|u|^2$. The square root of $| u|^2$ is obtained electronically and the result is integrated in the time domain to yield $\int udt$. A phase retarder located at plane P_{23} is used to invert the polarization of u, and appropriate attenuation is carried-out through a neutral density filter to yield $-\frac{1}{R}\int udt$. The resulting term, $-\frac{1}{R}\int udt$, is returned to the input of SLM_1, through the mirrors M_2 to yield the second term of Eq.(1.1).

The loop is energized by applying the initial entry vector v_{i0} through an array of laser diodes located at plane P_{41}. The number of diodes in the array is equal to number of elements of the vector$u$. The laser diodes can be interfaced with a computer to provide the vector v_0. The loop is energized after the initial entry vector v_0 is applied to the system through the laser diode arrays. The first cycle of the loop starts when v_0 is imaged onto plane P_{12} and a vector matrix product is performed between v_0 and T_{ij} elements stored on SLM_2. The result is imaged onto CCD_1 located at plane P_{13} and the intensity of $\sum v_{i0}T_{ij}$ is detected. The square root of $| \sum v_{i0}T_{ij}|^2$ is obtained electronically and the result is applied to SLM_3 located at plane P_{21}. The polarity of each element $\sum v_{i0}T_{ij}$ is determined optically by system L and the result is used to switch the pixels of SLM_3 accordingly. The vector elements$\sum v_{i0}T_{ij}$ are displayed on SLM_3 with their corresponding polarity and are imaged onto the input of SLM_1.

The local loop, generated by the beam splitter BS_1, provides u_0 which is detected by CCD_2. The square root of $| u_0|^2$ is obtained electronically; and the result is integrated in the time domain; appropriate scaling is carried out to yield $-\frac{1}{R}\int u_{i0}dt$ which is returned to the input of SLM_1. Thus, the first cycle of the operation is completed.

5. REFERENCES

[1] J. J. Hopfield, "neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci. U.S.A., vol. 79, pp. 2554- 2558, 1982.

[2] J. J. Hopfield, "neurons with graded response have collective computational properties like those of two-state neurons," Proc. Natl. Acad. Sci. U.S.A., vol. 81, pp. 3088- 3092, 1984.

[3] J. J. Hopfield and D. W. Tank, "neural computation of decisions optimization problems," Biological Cybern., vol. 52, pp. 141-152, 1985.

[4] J. J. Hopfield and D. W. Tank, "collective computation with continuous variables," in Disordered Systems and Biological Organization, E. Bienenstock, F. Fogelman, and G. Weisbush, eds., Springer-Verlag, 1985.

[5] J. J. Hopfield and D. W. Tank, "simple neural optimization networks: an A/D converter, a signal decision circuit, and a linear programming circuit," IEEE Trans. on Circuits and Systems, vol. CAS-33, no.5, May 1986, pp. 533-541.

[6] F. C. Hoppensteadt, an introduction to the mathematics of neurons, Cambridge University Press, 1986.

[7] Y. S. Abu-Mostafa and D. Pslatis, "Optical Neural Computers," Scientific American, 256, March 1987, pp. 88-95.

[8] R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP, April 1987, pp. 4-22.

[9] C. A. Mead, Analog VlSI and Neural Systems, Course Notes, Computer Science Dept., California Institute of Technology, 1986.

[10] M. P. Kennedy and L. O. Chua, "Neural Networks for Nonlinear Programming", IEEE CAS, to appear.

[11] E. R. Kandel and J. H. Schwartz, Principles of Neural Science, Elsevier, New York, 1985.

[12] P. E. Allen and D.R. Holberg, CMOS Analog Circuit Design, HRW, 1987.

[13] W. A. McCulloch and W. Pitts, Bull. Math. Biophys. 5, pp 115-133, 1943.

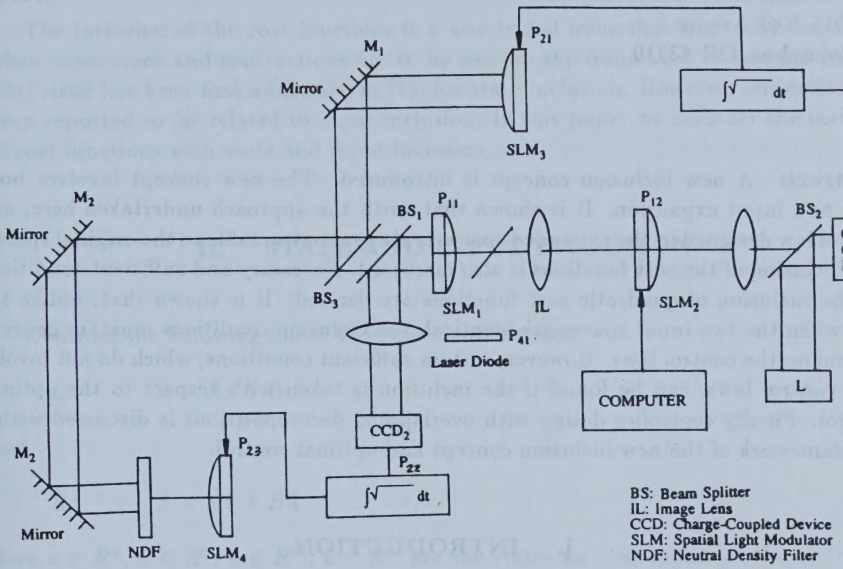
[14] W. P. Bleha et. al., "Application of the liquid Crystal Light Valve to Real-Time Optical Data Processing," Optical Engineering, vol. 17, No. 4 , 1978.

[15] B. Javidi and F. M. A. Salam, Real-Time Optical Neural Network," under preparation.

[16] M. Hirsch and S. Smale, Differential Equations, Dynamical Systems, and Linear Algebra, Academic Press, 1974.

[17] F M. A. Salam et. al., "The Theory of Simple Neural Nets," under preparation.

Figure 1. An Optical Neural Network.



P_{21}
M_1
Mirror
$\int\sqrt{\;}\,dt$
SLM_3
M_2
Mirror
BS_1
P_{11}
P_{12}
BS_2
P_{13}
C C D_1
BS_3
SLM_1
IL
SLM_2
P_{41}
Laser Diode
COMPUTER
L
CCD_2
P_{23}
P_{22}
M_2
$\int\sqrt{\;}\,dt$
Mirror
NDF
SLM_4
BS: Beam Splitter
IL: Image Lens
CCD: Charge-Coupled Device
SLM: Spatial Light Modulator
NDF: Neutral Density Filter

A NEW INCLUSION CONCEPT AND OPTIMAL CONTROL FOR SYSTEMS WITH OVERLAPPING INFORMATION SETS

ALTUĞ İFTAR and ÜMİT ÖZGÜNER
Department of Electrical Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210

Abstract: A new inclusion concept is introduced. The new concept involves both state and input expansion. It is shown that, with the approach undertaken here, any control law designed in the expanded spaces is always contractable to the original spaces. The inclusion of the cost functions is also discussed. Necessary and sufficient conditions for the inclusion of quadratic cost functions are derived. It is shown that, unlike the case when the two input spaces are identical, the inclusion conditions must, in general, depend on the control laws. However, certain sufficient conditions, which do not involve final control laws, can be found if the inclusion is taken with respect to the optimal control. Finally controller design with overlapping decompositions is discussed within the framework of the new inclusion concept and optimal control.

I. INTRODUCTION

Many of today's technological and social problems involve so complex systems that it is very costly if not impossible to handle such large dimensional systems as a whole. For the purposes of analysis and control, it is usually necessary to decompose the given system into a number of interconnected subsystems. Once a decomposition is available, each subsystem is to be considered independently and the solutions to the subproblems are to be combined to obtain a solution for the original problem.

In order to obtain a useful decomposition, it is essential to identify the parts of a system that are *weakly* interconnected. However, many large scale systems may consist of subsystems which are strongly connected through certain dynamics (*"the overlapping part"*), but weakly connected otherwise. Electric power systems [1], socio-economic systems [2], large flexible structures [3], and freeway traffic regulation [4] are examples of such behaviour. For such systems, disjoint decompositions may easily fail to produce useful results. However, it has been demonstrated that the recently introduced *overlapping decompositions* [5] may produce useful solutions in such cases [5]-[15].

The controller design approach within the framework of the *Inclusion Principle* [9] starts with expanding certain spaces (ie. state, input *etc.*) of a dynamic system with overlapping subsystems into larger spaces such that the subsystems appear as disjoint. Then, decentralized control laws are designed in the expanded spaces and finally contracted to the smaller spaces for implementation on the original system.

With the exception of a few papers [13,14], all of the results so far have been related to the expansions and contractions of the state space only. In this paper, we address expansions and contractions of the input space as well as the state space. We introduce a new concept, called *extension.* It is shown that, if this new concept is employed, any control law designed in the expanded spaces can always be contracted to the original spaces.

The inclusion of the cost functions is a non-trivial issue that has to be considered when expansions and contractions are to be used in the framework of optimal control. This issue has been first addressed in [15] for state inclusion. However, no results have been reported so far related to input inclusion. In this paper we consider the inclusion of cost functions with state and input inclusion.

II. STATE AND INPUT INCLUSION

Consider the following linear time-invariant systems:

$$\Sigma: \qquad \dot{x} = Ax + Bu \tag{1}$$

and

$$\tilde{\Sigma}: \qquad \dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}\tilde{u} \tag{2}$$

Here $x \in R^n$, $\tilde{x} \in R^{\tilde{n}}$, $u \in R^m$, $\tilde{u} \in R^{\tilde{m}}$ are the states and the inputs of the systems Σ and $\tilde{\Sigma}$ respectively. It is assumed that $\tilde{n} \geq n$ and $\tilde{m} \geq m$.

Consider the transformations:

$$T: R^n \rightarrow R^{\tilde{n}} \quad , \qquad rank(T) = n \tag{3a}$$

$$U: R^{\tilde{m}} \rightarrow R^m \quad , \qquad rank(U) = m \tag{3b}$$

$$T^{\#}: R^{\tilde{n}} \rightarrow R^n \quad , \qquad T^{\#}T = I_n \tag{3c}$$

$$U^{\#}: R^m \rightarrow R^{\tilde{m}} \quad , \qquad UU^{\#} = I_m \tag{3d}$$

where I_k is the identity operator on R^k.

Definition: The system $\tilde{\Sigma}$ is an *extension* of the system Σ, and Σ is a *disextension* of $\tilde{\Sigma}$, if there exist transformations as in (3) such that for any initial state $x_o \in R^n$ and any fixed input $\tilde{u}(t) \in R^{\tilde{m}}$, $0 \leq t < \infty$, the choice

$$\tilde{x}_o = Tx_o \tag{4a}$$

$$u(t) = U\tilde{u}(t) \quad \forall t \geq 0 \tag{4b}$$

implies that

$$\tilde{x}(t;\tilde{x}_o,\tilde{u}) = Tx(t;x_o,u) \quad \forall t \geq 0. \tag{5}$$

The extension defined above is a special case of inclusion. In fact, it is a generalization of *unrestriction*, defined in [9], to the case of state and input inclusion. However, it is different than the unrestriction defined in [13]. In [13] the unrestriction is defined for an arbitrary input $u(t)$ in the original input space, and the input in the expanded space is obtained by a transformation: $\tilde{u}(t) = U^{\#}u(t)$. Here the extension is defined for an arbitrary input $\tilde{u}(t)$ in the expanded input space, and the input in the original space is obtained by the contraction given in (4b). Therefore, for the unrestriction the allowable set of inputs for $\tilde{\Sigma}$ at any time is only an m-dimensional subset of $R^{\tilde{m}}$, but for the extension it is $R^{\tilde{m}}$. However, this additional freedom brings in certain additional restrictions in the inclusion conditions which are provided by the following:

Theorem 1: The system $\tilde{\Sigma}$ is an extension of the system Σ if and only if there exist transformations as in (3) such that

$$TA = \tilde{A}T \quad , \qquad TBU = \tilde{B} \quad . \tag{6}$$

Under the transformations (3), the matrices of Σ and $\tilde{\Sigma}$ can be related as:

$$\tilde{A} = TAT^{\#} + M \quad , \qquad \tilde{B} = TBU + N \tag{7}$$

where M and N are constant complementary matrices. In order $\tilde{\Sigma}$ to be an extension of Σ, these complementary matrices must satisfy certain conditions:

Theorem 2: The system $\tilde{\Sigma}$ is an extension of the system Σ if and only if

$$MT = 0 \quad , \qquad N = 0. \tag{8}$$

Next we consider the linear feedback laws:

$$u = Kx \tag{9}$$

for the system Σ, and

$$\tilde{u} = \tilde{K}\tilde{x} \tag{10}$$

for $\tilde{\Sigma}$.

Definition: The control law (10) is *contractable* to the control law (9), if there exist transformations as in (3a) and (3b) such that

$$Kx(t;x_o,U\tilde{u}) = U\tilde{K}\tilde{x}(t;Tx_o,\tilde{u}) \quad ,\forall t \geq 0 \tag{11}$$

for any initial state $x_o \in R^n$ and any fixed input $\tilde{u}(t) \in R^{\tilde{m}}$, $0 \leq t < \infty$.

Note that, if (10) is contractable to (9), then the closed loop system

$$\tilde{\Sigma}_F : \qquad \dot{\tilde{x}} = (\tilde{A} + \tilde{B}\tilde{K})\tilde{x} \tag{12}$$

is an extension of the closed loop system:

$$\Sigma_F : \qquad \dot{x} = (A + BK)x \ . \tag{13}$$

The two feedback gain matrices in (9) and (10) can be related by:

$$\tilde{K} = U^{\#} K T^{\#} + L \tag{14}$$

where L is a constant complementary matrix, which must satisfy the following conditions in order for the contractability to hold.

Theorem 3: The control law (10) is contractable to the control law (9) if and only if:

$$ULM^{i-1}T = 0 \ , \qquad ULM^{i-1}N = 0 \ , \qquad \forall i = 1, 2, ..., \tilde{n} \tag{15}$$

Since the control laws are to be designed in the expanded spaces and then contracted for implementation, it is important that any control law designed in the expanded spaces be contractable. In fact, it follows from Theorems 2 and 3 that if $\tilde{\Sigma}$ is an extension of Σ then such a property holds.

Corollary 1: If $\tilde{\Sigma}$ is an extension of Σ then any control law of the form (10) is contractable to a control law of the form (9), and K is given by:

$$K = U\tilde{K}T \ . \tag{16}$$

III. INCLUSION OF COST FUNCTIONS

Consider the cost functions:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{17}$$

and

$$\tilde{J} = \int_0^\infty (\tilde{x}^T \tilde{Q} \tilde{x} + \tilde{u}^T \tilde{R} \tilde{u}) dt \tag{18}$$

for the systems Σ and $\tilde{\Sigma}$ respectively. The inclusion of cost functions has been first discussed in [15] for the case of state inclusion only. Direct extension of the definition given in [15] to state and input inclusion is as follows:

Definition: The pair $(\tilde{\Sigma}, \tilde{J})$ *includes* the pair (Σ, J) if there exist transformations as in (3), such that $\tilde{\Sigma}$ *includes* Σ [13] and

$$J(x_o, U\tilde{u}) = \tilde{J}(Tx_o, \tilde{u}) \tag{19}$$

for any initial state $x_o \in R^n$ and any fixed input $\tilde{u}(t) \in R^{\tilde{m}}$, $0 \leq t < \infty$.

It can easily be verified that the control weighting matrix in (18) must satisfy:

$$rank(\tilde{R}) \leq m$$

for a valid inclusion. This requires a non-positive-definite control weighting matrix for the expanded input whenever $\tilde{m} > m$. This property is certainly undesirable since the control law designed in the expanded spaces would be improper. To overcome this difficulty, we introduce a new inclusion concept as follows:

Definition: Let K and $\tilde{K}$ be such that $A + BK$ and $\tilde{A} + \tilde{B}\tilde{K}$ are asymptotically stable. Then the pair $(\tilde{\Sigma}, \tilde{J})$ *includes* the pair (Σ, J) *with respect to the feedback control laws* $(Kx, \tilde{K}\tilde{x})$ if there exist transformations as in (3) such that $\tilde{\Sigma}$ includes Σ, $\tilde{K}\tilde{x}$ is contractable to Kx, and

$$J(x_o, Kx) = \tilde{J}(Tx_o, \tilde{K}\tilde{x}) \tag{20}$$

for any initial state $x_o \in R^n$.

Under the transformations (3), the matrices of J and $\tilde{J}$ can be related as:

$$\tilde{Q} = (T^{\#})^T Q T^{\#} + M_Q \quad , \qquad \tilde{R} = U^T R U + N_R \tag{21}$$

where M_Q and N_R are constant complementary matrices.

Now let $\tilde{\Sigma}$ be an extension of Σ. Then, by Corollary 1, any feedback control law of the form (10) is contractable to a feedback control law of the form (9), and contracted feedback gains are given by (16). Under these assumptions, the necessary and sufficient conditions for the inclusion of cost functions are given by the following:

Theorem 4: If $\tilde{\Sigma}$ is an extension of Σ, $K = U\tilde{K}T$, and $\tilde{K}$ is such that $\tilde{A} + \tilde{B}\tilde{K}$ is asymptotically stable, then the pair $(\tilde{\Sigma}, \tilde{J})$ includes the pair (Σ, J) with respect to $(Kx, \tilde{K}\tilde{x})$ if and only if

$$T^T (M_Q + \tilde{K}^T N_R \tilde{K}) T = 0 \ . \tag{22}$$

By using the above theorem, it is always possible to construct a positive definite $\tilde{R}$ and a positive semi-definite $\tilde{Q}$ for given positive definite R and positive semi-definite Q. However, the construction depends on the feedback gain matrix $\tilde{K}$. This dependence is in general undesirable, since the feedback gains to be designed depend on the choice of $\tilde{R}$ and $\tilde{Q}$. This difficulty can be resolved if the inclusion is defined with respect to the optimal control.

Definition: Suppose that there exists a bounded control of the form (10) such that $\tilde{J}(\tilde{x}_o, \tilde{u})$ is minimized for all $\tilde{x}_o \in R^{\tilde{n}}$. Then the pair $(\tilde{\Sigma}, \tilde{J})$ *includes* the pair (Σ, J) *with respect to the optimal* if $\tilde{\Sigma}$ includes Σ, and

$$\min_u J(x_o, u) = \min_{\tilde{u}} \tilde{J}(Tx_o, \tilde{u}) \tag{23}$$

for any initial state $x_o \in R^n$.

Theorem 5: If $\tilde{\Sigma}$ is an extension of Σ and there exists a bounded control of the form (10) such that $\tilde{J}(\tilde{x}_o, \tilde{u})$ is minimized for any $\tilde{x}_o \in R^{\tilde{n}}$, then the pair $(\tilde{\Sigma}, \tilde{J})$ includes the pair (Σ, J) with respect to the optimal if

$$T^T M_Q T = 0 \quad , \qquad (U^{\#})^T N_R U^{\#} = 0 \ . \tag{24}$$

Note that, it is always possible to construct symmetric matrices M_Q and N_R satisfying (24) such that $\tilde{Q}$ and $\tilde{R}$ defined by (21) are positive semi definite and positive definite respectively. In fact, one may as well start with positive semi definite and positive definite matrices $\tilde{Q}$ and $\tilde{R}$ and then construct Q and R by:

$$Q = T^T \tilde{Q} T \tag{25}$$

and

$$R = (U^{\#})^T \tilde{R} U^{\#} \ . \tag{26}$$

IV. OVERLAPPING DECOMPOSITIONS

Consider the system Σ given in (1). Suppose that the state and the input are partitioned as:

$$x = (x_1^T \ , \ x_2^T \ , \ x_3^T)^T \quad , \qquad x_i \in R^{n_i} \ , \tag{27a}$$

$$u = (u_1^T \ , \ u_2^T \ , \ u_3^T)^T \quad , \qquad u_i \in R^{m_i} \ . \tag{27b}$$

Here it is assumed that x_2 and u_2 correspond to the overlapping parts of the state and input spaces respectively. Suppose the system matrices in (1) are also partitioned compatibly:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \ , \qquad B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \ . \tag{28}$$

An extension of Σ is obtained by choosing

$$T = \begin{bmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ 0 & I_{n_2} & 0 \\ 0 & 0 & I_{n_3} \end{bmatrix} \ , \qquad U = \begin{bmatrix} I_{m_1} & 0 & 0 & 0 \\ 0 & I_{m_2} & I_{m_2} & 0 \\ 0 & 0 & 0 & I_{m_3} \end{bmatrix} \ .$$

$$T^{\#} = \begin{bmatrix} I_{n_1} & 0 & 0 & 0 \\ 0 & \frac{1}{2} I_{n_2} & \frac{1}{2} I_{n_2} & 0 \\ 0 & 0 & 0 & I_{n_3} \end{bmatrix} \ , \quad M = \begin{bmatrix} 0 & \frac{1}{2} A_{12} & -\frac{1}{2} A_{12} & 0 \\ 0 & \frac{1}{2} A_{22} & -\frac{1}{2} A_{22} & 0 \\ 0 & -\frac{1}{2} A_{22} & \frac{1}{2} A_{22} & 0 \\ 0 & -\frac{1}{2} A_{32} & \frac{1}{2} A_{32} & 0 \end{bmatrix} \ ,$$

$$N = 0 \ . \tag{29}$$

The resulting expanded system matrices are:

$$\tilde{A} = \begin{bmatrix} A_{11} & A_{12} & \vdots & 0 & A_{13} \\ A_{21} & A_{22} & \vdots & 0 & A_{23} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{21} & 0 & \vdots & A_{22} & A_{23} \\ A_{31} & 0 & \vdots & A_{32} & A_{33} \end{bmatrix} \triangleq \begin{bmatrix} \tilde{A}_1 & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_2 \end{bmatrix} , \tag{30a}$$

$$\tilde{B} = \begin{bmatrix} B_{11} & B_{12} & \vdots & B_{12} & B_{13} \\ B_{21} & B_{22} & \vdots & B_{22} & B_{23} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{21} & B_{22} & \vdots & B_{22} & B_{23} \\ B_{31} & B_{32} & \vdots & B_{32} & B_{33} \end{bmatrix} \triangleq \begin{bmatrix} \tilde{B}_1 & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_2 \end{bmatrix} . \tag{30b}$$

Next, consider the *decoupled subsystems*:

$$\tilde{\Sigma}_i^D : \qquad \dot{\tilde{x}}_i = \tilde{A}_i \tilde{x}_i + \tilde{B}_i \tilde{u}_i \quad , \qquad i = 1, 2. \tag{31}$$

and corresponding cost functions:

$$\tilde{J}_i = \int_0^\infty (\tilde{x}_i^T \tilde{Q}_i \tilde{x}_i + \tilde{u}_i^T \tilde{R}_i \tilde{u}_i) dt \quad , \qquad i = 1, 2 \tag{32}$$

The total cost function for the expanded system is:

$$\tilde{J} = \tilde{J}_1 + \tilde{J}_2 = \int_0^\infty (\tilde{x}^T \tilde{Q} \tilde{x} + \tilde{u}^T \tilde{R} \tilde{u}) dt \tag{33}$$

where

$$\tilde{Q} = blockdiag(\tilde{Q}_1 \ , \ \tilde{Q}_2) \quad , \qquad \tilde{R} = blockdiag(\tilde{R}_1 \ , \ \tilde{R}_2) \quad . \tag{34}$$

Consider the cost function (17) for Σ. $(\tilde{\Sigma}, \tilde{J})$ includes (Σ, J), with respect to the optimal if Q and R satisfy (25) and (26), respectively, for some $U^\#$ satisfying $UU^\# = I_m$. In particular if:

$$Q = blockdiag(Q_1 \ , \ Q_2 \ , \ Q_3) \quad , \qquad R = blockdiag(R_1 \ , \ R_2 \ , \ R_3) \tag{35}$$

where $Q_i \in R^{n_i \times n_i}$, $R_i \in R^{m_i \times m_i}$, and $U^\#$ is chosen as:

$$U^\# = U^T (UU^T)^{-1} \quad , \tag{36}$$

then $\tilde{Q}_i$ and $\tilde{R}_i$ $(i = 1, 2)$ may be chosen as follows:

$$\tilde{Q}_1 = blockdiag(Q_1 \ , \ \tilde{Q}_{2_1}) \quad , \qquad \tilde{Q}_2 = blockdiag(\tilde{Q}_{2_2} \ , \ Q_3) \tag{37}$$

where $\tilde{Q}_{2_1} \geq 0$, $\tilde{Q}_{2_2} \geq 0$, and $\tilde{Q}_{2_1} + \tilde{Q}_{2_2} = Q_2$;

$$\tilde{R}_1 = blockdiag(R_1 \ , \ \tilde{R}_{2_1}) \quad , \qquad \tilde{R}_2 = blockdiag(\tilde{R}_{2_2} \ , \ R_3) \tag{38}$$

where $\tilde{R}_{2_1} > 0$, $\tilde{R}_{2_2} > 0$, and $\tilde{R}_{2_1} + \tilde{R}_{2_2} = 4R_2$. In particular one may let:

$$\tilde{Q}_{2_1} = \tilde{Q}_{2_2} = \frac{1}{2}Q_2 \quad , \qquad \tilde{R}_{2_1} = \tilde{R}_{2_2} = 2R_2 \quad . \tag{39}$$

Now, suppose that the control laws

$$\tilde{u}_i = \tilde{K}^i \tilde{x}_i \quad , \qquad i = 1, 2 \tag{40}$$

are designed (say to minimize $\tilde{J}_i$) for the decoupled subsystems $\tilde{\Sigma}_i^D$, for $i = 1, 2$, respectively. Partition $\tilde{K}^i$ as follows:

$$\tilde{K}^i = \begin{bmatrix} \tilde{K}^i_{11} & \tilde{K}^i_{12} \\ \tilde{K}^i_{21} & \tilde{K}^i_{22} \end{bmatrix} \tag{41}$$

where $\tilde{K}^1_{11} \in R^{m_1 \times n_1}$; $\tilde{K}^1_{22}$, $\tilde{K}^2_{11} \in R^{m_2 \times n_2}$; $\tilde{K}^2_{22} \in R^{m_3 \times n_3}$. Then the feedback control law:

$$\tilde{u} = \tilde{K}\tilde{x} \quad , \qquad \tilde{K} \triangleq blockdiag(\tilde{K}^1 \, , \, \tilde{K}^2) \tag{42}$$

is contracted to the original spaces as:

$$u = Kx \tag{43}$$

where

$$K = \begin{bmatrix} \tilde{K}^1_{11} & \tilde{K}^1_{12} & 0 \\ \tilde{K}^1_{21} & \tilde{K}^1_{22} + \tilde{K}^2_{11} & \tilde{K}^2_{12} \\ 0 & \tilde{K}^2_{21} & \tilde{K}^2_{22} \end{bmatrix} \tag{44}$$

The implementation of this control strategy is summarized in figure 1.

V. CONCLUSIONS

A new inclusion concept has been introduced. The new concept involves both state and input expansion and is conceptually close to the unrestriction defined in [13]. However, in [13] the unrestriction was defined for an arbitrary input in the original input space, and the input in the expanded space was obtained by a transformation. This fact brings in the restriction that a particular controller designed for the expanded system by the approach of [13] may not be contractable to the original spaces. It has been shown, however, that with the approach undertaken here any control law designed in the expanded spaces is always contractable to the original spaces. On the other hand, given the original system, there is, in general, more freedom of choosing the input matrix $\tilde{B}$ for the expansion, if unrestriction is employed rather than extension. Therefore, it is possible to obtain an expanded system which has subsystems with less interaction. However, such low interaction in $\tilde{B}$ is at the end compensated by the high interaction in the controller, that has to be introduced in order for the contractability to hold.

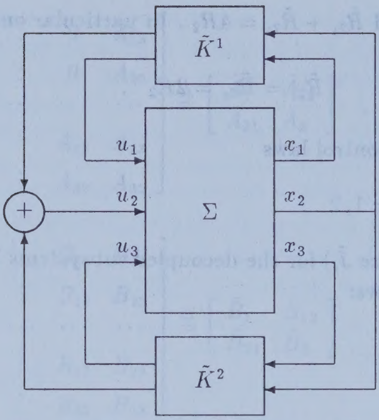


Figure 1: Control Law Implementation

The inclusion of the cost functions has also been discussed. Necessary and sufficient conditions for the inclusion of quadratic cost functions have been derived. It has been shown that, unlike the case when the two input spaces are identical, the inclusion conditions must, in general, depend on the control laws. However, certain sufficient conditions, which do not involve final control laws, can be found if the inclusion is taken with respect to the optimal control.

Finally controller design with overlapping decompositions has been discussed within the framework of the new inclusion concept and optimal control. Although systems with only two subsystems have been presented for notational simplicity, the extension of the results to systems with more than two subsystems is straightforward.

ACKNOWLEDGMENT

The authors would like to thank Professor Dragoslav D. Šiljak for helpful discussions.

REFERENCES

[1] D. D. Šiljak, *Large-Scale Dynamic Systems: Stability and Structure*. New York: North-Holland, 1978.

[2] M. Aoki, "On decentralized stabilization and dynamic assignment problems," *J. International Economics*, vol. 6, pp. 143–171, 1976.

[3] K. D. Young, "An application of decomposition techniques to control of large structures," in *Proc. Fourth VPI&SU/AIAA Symposium*, pp. 201–214, Blacksburg, VA, June 1983.

[4] L. Isaksen and H. J. Payne, "Suboptimal control of linear systems by augmentation with application to freeway traffic regulation," *IEEE Transactions on Automatic Control*, vol. AC–18, pp. 210–219, 1973.

[5] M. Ikeda and D. D. Šiljak, "Overlapping decompositions, expansions, and contractions of dynamic systems," *Large scale systems*, vol. 1, pp. 29–38, 1980.

[6] M. Ikeda and D. D. Šiljak, "Generalized decompositions of dynamic systems and vector Liapunov functions," *IEEE Transactions on Automatic Control*, vol. AC–26, pp. 1118–1125, 1981.

[7] Y. Ohta and D. D. Šiljak, "An inclusion principle for hereditary systems," *J. Mathematical analysis and Applications*, vol. 98, pp. 581–598, 1984.

[8] M. E. Sezer and D. D. Šiljak, "Validation of reduced order models for control systems design," *Journal of Guidance and Control*, vol. 5, pp. 430–437, 1982.

[9] M. Ikeda, D. D. Šiljak, and D. E. White, "An inclusion principle for dynamic systems," *IEEE Transactions on Automatic Control*, vol. AC–29, pp. 244–249, 1984.

[10] A. İftar and Ü. Özgüner, "Closed-loop balanced realizations in the analysis of suboptimality and stability of decentralized systems," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 143–148, Las Vegas, Nevada, December 1984.

[11] A. İftar and Ü. Özgüner, "Decentralized LQG/LTR controller design for interconnected systems," in *Proceedings of the American Control Conference*, pp. 1682–1687, Minneapolis, MN, June 1987.

[12] A. İftar and Ü. Özgüner, "Local LQG/LTR controller design for decentralized systems," *IEEE Transactions on Automatic Control*, vol. AC–32, , October 1987. (to appear).

[13] M. Ikeda and D. D. Šiljak, "Overlapping decentralized control with input, state, and output inclusion," *Control Theory and Advanced Technology*, vol. 2, pp. 155–172, 1986.

[14] Y. Ohta, D. D. Šiljak, and T. Matsumoto, "Decentralized control using quasi-block diagonal dominance of transfer function matrices," *IEEE Transactions on Automatic Control*, vol. AC–31, pp. 420–430, 1986.

[15] M. Ikeda, D. D. Šiljak, and D. E. White, "Decentralized control with overlapping information sets," *J. Optimization Theory and Applications*, vol. 34, pp. 279–310, 1981.

DISTRIBUTED COMPUTATIONS OF OPTIMAL CONTROL STRATEGIES FOR A CLASS OF INTERCONNECTED SYSTEMS

LEVENT ACAR *and* ÜMIT ÖZGÜNER
Department of Electrical Engineering
The Ohio State University
2015 Neil Ave.
Columbus, OH 43210

Abstract: In this paper, distributed computations of optimal control strategies for moderately coupled interconnected systems are examined. Distributed computations are well suited for interconnected systems for two reasons. First, interconnected systems have natural computational decompositions. Second, decentralized optimal finite horizon linear quadratic regulator problem can not be solved analytically for interconnected systems and possible sequential computations may require extensive computational time. Two different computational decompositions and two corresponding iterative methods are proposed to compute optimal feedback gains. Convergence proofs of these methods are presented in the extention of this work. The performances of the methods are compared in an example.

I. INTRODUCTION

Recent developments in multi-processing systems introduced new methods to solve large and distributed problems [1]–[3]. One such problem is the optimal control of large interconnected systems.

In interconnected systems, the physical existence of subsystems connected by interactions gives a natural decomposition of work load among individual processors. Although the essential distribution of the computational load can be easily divided, the details of data transfer between the processors may vary from one application to another.

We propose two different strategies to organize the communications among processors to solve the finite horizon decentralized optimal control problem for discrete systems. The first strategy is to obtain the most recent and updated information as soon as possible. This strategy demands massive data transfers among processors. The second strategy is to obtain minimal information and use it to accomplish the required task. In this strategy, communications are minimized, but the load work of each processor is increased.

We also present two iterative computational methods implementing these strategies. The methods have a linear convergence rate as a function of the interaction size of the subsystems. We also included an example to compare the performances of the two strategies.

II. FORMULATION

In this section, we present the necessary equations for optimality of a finite horizon cost criterion under decentralized implementation constraints. We consider a linear, discrete-time, composite system, such that each subsystem is described by the equation

$$x_i(k+1) = A_{ii}(k)x_i(k) + \sum_{\substack{j=1 \\ j\neq i}}^{p} A_{ij}(k)x_j(k) + B_i(k)u_i(k) \tag{1}$$

where $x_i \in \mathcal{R}^{n_i}$, $A_{ij} \in \mathcal{R}^{n_i} \times \mathcal{R}^{n_j}$, $B_i \in \mathcal{R}^{n_i} \times \mathcal{R}^{m_i}$ and $u_i \in \mathcal{R}^{m_i}$ for $i = 1, \ldots, p$. Here, x_i and u_i represent the state vector and the local input vector of the i'th subsystem. We consider a decentralized control of the form

$$u_i(k) = F_i(k)x_i(k) \tag{2}$$

which minimizes a quadratic cost function

$$J = \frac{1}{2}\sum_{i=1}^{p}\left\{ x_i^T(N)H_ix_i(N) + \sum_{k=0}^{N-1}\left[x_i^T(k)Q_i(k)x_i(k) + u_i^T(k)R_i(k)u_i(k)\right]\right\} \tag{3}$$

where N is the specified horizon length, $F_i \in \mathcal{R}^{m_i} \times \mathcal{R}^{n_i}$ and H_i, Q_i, and R_i are symmetric matrices with compatible dimensions.

Optimal control equations for the above team decision problem is determined in [4] by using the method of dynamic programming [5]. In presenting these equations, we will suppress the explicit time step parameters for the sake of notational brevity. We will use " + " to denote the updated values of P and x, and other parameters will be totally suppressed. Also, the variables i, j and k will always denote subsystems unless explicitly specified elsewise.

The update expressions for the cost updating matrices are determined as follows

$$P_{ii}(+) = Q_i + F_i^T(R_i + B_i^T P_{ii} B_i)F_i$$

$$+ \sum_{j=1}^{p}\left[A_{ji}^T P_{ji} B_i F_i + F_i^T B_i^T P_{ij} A_{ji}\right] + \sum_{j_1=1}^{p}\sum_{j_2=1}^{p} A_{j_1 i}^T P_{j_1 j_2} A_{j_2 i}\,, \tag{4}$$

and

$$P_{i_1 i_2}(+) = F_{i_1}^T B_{i_1}^T P_{i_1 i_2} B_{i_2} F_{i_2} + \sum_{j=1}^{p}\left[A_{j i_1}^T P_{j i_2} B_{i_2} F_{i_2} + F_{i_1}^T B_{i_1}^T P_{i_1 j} A_{j i_2}\right]$$

$$+ \sum_{j_1=1}^{p}\sum_{j_2=1}^{p} A_{j_1 i_1}^T P_{j_1 j_2} A_{j_2 i_2} \tag{5}$$

for $i_1 \neq i_2$. For symmetric P_{ii} and $P_{ij} = P_{ji}^T$, the optimal feedback gains satisfy

$$\left[R_iF_ix_i + \sum_{j_1=1}^{p} B_i^T\left(P_{ij_1}B_{j_1}F_{j_1} + \sum_{j_2=1}^{p} P_{ij_2}A_{j_2j_1}\right)x_{j_1}\right]x_i^T = 0 \tag{6}$$

for all $i = 1, \ldots, p$. From the above optimality condition, we realize that the optimal feedback gains cannot be determined independent of the initial states in general.

The couplings introduced by the state dependence and the two end boundary values of the equations (4), (5) and (6) eliminate an analytical solution approach. Some suboptimal control methods are used to solve coupled system of equations [4], [7].

III. METHODS

In this section, we explore possibilities of approximating the optimal solution by two iterative methods especially designed for distributed processing. The processing load is divided such that each processor is assigned to determine the feedback gains of one of the subsystems. Hence, we assume we have at least p processors.

Similar to [4], we assume that an interconnection parameter ρ; $0 \leq \rho < 1$ is extracted from the interconnection matrices A_{ij} such that

$$A_{ij} = \rho\delta A_{ij} \; ; \quad j = 1, \ldots, p, \quad j \neq i. \tag{7}$$

Here, A_{ij}'s and A_{ii} are assumed to be comparable in norm so that the interactions are not dominating the subsystems [1]. We follow the same notation used in [4] and use “ $\tilde{\cdot}$ ” for the nominal portions and “ $\delta\cdot$ ” for the interconnection portions. As a result, we define

$$\begin{array}{ll||ll} A_{ii} = \tilde{A}_i \; , & & & F_i = \tilde{F}_i + \rho\delta F_i \; , \\ x_i = \tilde{x}_i + \rho\delta x_i \; , & & & P_{ii} = \tilde{P}_i + \rho\delta P_i \; , \end{array} \tag{8}$$

and

$$P_{ij} = \rho\delta P_{ij}$$

for all $i, j = 1, \ldots, p; j \neq i$. By substituting (7) and (8) into the necessary conditions for optimality, the optimal feedback gain equations of the nominal and interconnection portions can be derived [4].

To derive a more general and applicable decentralized control strategy, we do not assume the exact values of the initial states to be known. Instead we assume the initial states have random distributions and consider the expected value of the cost function (3). The cross-covariance distribution matrices are defined as follows:

$$\begin{array}{ll} \chi_{ij} = \mathrm{E}\{\tilde{x}_i \tilde{x}_j^T\} \; , & \\ & \text{and} \qquad \delta^2\chi_{ij} = \mathrm{E}\{\delta x_i \delta x_j^T\} \; . \\ \delta\chi_{ij} = \mathrm{E}\{\delta x_i \tilde{x}_j^T\} \; , & \end{array} \tag{9}$$

where $\mathrm{E}\{\cdot\}$ is the expected value operator. The resulting update equations of these covariance matrices can then be obtained. If the state covariance matrices χ_{ij} are also not known, then we will assume the variances are located on a unit ball where $\chi_{ii}(0)$ is identity and $\chi_{ij}(0)$ is zero for all distinct i and j.

---

[1] This condition is easily achieved, when $||\delta A_{ij}|| \leq 1$, where $||\cdot||$ represents an induced matrix norm, and initial conditions are comparable.

In determining the optimal solution, each subsystem needs to know other subsystems' feedback gains. The methods proposed in this work differ in the use of this information.

The first method iteratively approximates the complete set of coupled equations. First, state cross-covariance matrices are computed using an initial set of feedback gains. Second, new feedback gains are computed by utilizing these covariance matrices. Finally, the new feedback gains are transmitted to other processors to help them determine their cost updating matrices. This procedure repeats until all the feedback gains are computed and some convergence criteria are achieved. We call this type of computation the *team processing*.

The second method uses a more individualistic strategy. Instead of massive information transfers between processors, each time a processor receives a set of feedback gains from other subsystems, the processor calculates its own feedback gains independent of any intermediate calculations of other processors. The cost updating and the covariance matrices are all locally computed and each processor uses a different set of updating matrices. We call this type of computation the *individual processing*.

The computational procedures of both of the processes start with calculations of the nominal variables. In this phase, feedback gains are independent of the state variables. As a result, each processor computes its nominal feedback gains through out the horizon independently. To divide the computational load evenly, we also separate cost updating matrices δP_{ij} and the covariance matrices χ_{ij} and $\delta^2 \chi_{ij}$ into two components. We assume

$$\begin{aligned} &\delta P_{ij} = \delta P^i_{ij} + \delta P^{j^T}_{ji} , \\ &\qquad\qquad\qquad\qquad \text{and} \qquad \delta^2 \chi_{ij} = \delta^2 \chi^i_{ij} + \delta^2 \chi^{j^T}_{ji} , \\ &\chi_{ij} = \chi^i_{ij} + \chi^{j^T}_{ji} , \end{aligned} \tag{10}$$

where supercripts denote the processors computing that component of the variable. The detailed computational procedure of this phase is given below.

Team and Individual Processing: Initial Phase

Data Status
- Received: None
- Available: $\chi_{ii}(0)$, $\chi^i_{ij}(0)$; for all $j \neq i$
- Transmitted: $(\tilde{A}_i + B_i \tilde{F}_i(0)), \ldots, (\tilde{A}_i + B_i \tilde{F}_i(N-1))$, $\tilde{P}_i(0), \ldots, \tilde{P}_i(N)$, $\chi_{ii}(1), \ldots, \chi_{ii}(N)$ and $\chi^i_{ij}(1), \ldots, \chi^i_{ij}(N)$; for all $j \neq i$

1. Determine $\tilde{F}_i(N-1)$, $\tilde{P}_i(1), \ldots, \tilde{F}_i(0)$, $\tilde{P}_i(N)$ recursively from $\tilde{F}_i = -(R_i + B_i^T \tilde{P}_i B_i)^{-1} \cdot B_i^T \tilde{P}_i \tilde{A}_i$ and $\tilde{P}_i(+) = (\tilde{A}_i + B_i \tilde{F}_i)^T \tilde{P}_i (\tilde{A}_i + B_i \tilde{F}_i) + \tilde{F}_i^T R_i \tilde{F}_i + Q_i$ where $\tilde{P}_i(0) = H_i$.
2. Determine $(\tilde{A}_i + B_i \tilde{F}_i(0)), \ldots, (\tilde{A}_i + B_i \tilde{F}_i(N-1))$, and $\chi_{ii}(1)$, $\chi^i_{ij}(1), \ldots, \chi_{ii}(N)$ $\chi^i_{ij}(N)$ for all $j \neq i$ from the results obtained in step 1 and from $\chi_{ii}(+) = (\tilde{A}_i + B_i \tilde{F}_i) \chi_{ii} (\tilde{A}_i + B_i \tilde{F}_i)^T$ and $\chi^i_{ij}(+) = (\tilde{A}_i + B_i \tilde{F}_i) \chi^i_{ij} (\tilde{A}_j + B_j \tilde{F}_j)^T$.

Next step is to compute the interconnection portions of the feedback gains. This step of the *team processing* strategy has two phases. First phase is the computations of the state covariance matrices and the second phase is the computations of the feedback gains using the already computed state covariances. We first give the global procedure for the *team processing* and then describe the two phases.

Team Processing: Global

1. Start from initial time and calculate all the state covariance matrices using the forward recursion equations in *Phase I* described below.
2. Determine the trace of the cost function and if the difference between the current value of the cost and the value obtained from the previous iteration is small, the procedure ends.
3. Start from the end of the horizon and calculate interconnection portions of the feedback gains and the cost updating matrices using the backward recursion equations in *Phase II* described below.
4. Go back to step 1.

Team Processing: Phase I

Data Status
- Received: $\delta\chi_{jk}$, $\delta^2\chi_{jj}$ and $\delta^2\chi^j_{jk}$; for all $j \neq i$, $k \neq j$,
- Available: $\delta\chi_{ik}$, $\delta^2\chi_{ii}$ and $\delta^2\chi^i_{ik}$; for all k
- Transmitted: $\delta\chi_{ik(+)}$, $\delta^2\chi_{ii(+)}$ and $\delta^2\chi^i_{ij(+)}$; for all $j \neq i$ and k

1. Determine $\delta\chi_{ik(+)}$ from (12) where $\delta\chi_{ik(0)}$ is zero.
2. Determine $\delta^2\chi_{ii(+)}$, and $\delta^2\chi^i_{ij(+)}$ from (13) and $\delta^2\chi_{ii(+)} = \left[\delta^2\chi^i_{ij(+)} + \delta^2\chi^{jT}_{ji}{(+)}\right]_{j=i}$ where $\delta^2\chi_{ii(0)}$ and $\delta^2\chi^i_{ij(0)}$ are zero.

Team Processing: Phase II

Data Status
- Received: $\delta F^{(-)}_j$, δP_j, δP^j_{jk}, $\delta\chi_{jk}$, $\delta^2\chi_{jj}$, $\delta^2\chi^j_{jk}$; for all $j \neq i$ and $k \neq j$
- Available: $\delta F^{(-)}_i$, δP_i, δP^i_{ij}, $\delta\chi_{ik}$, $\delta^2\chi_{ii}$ and $\delta^2\chi^i_{ij}$; for all $j \neq i$ and k
- Transmitted: δF_i, $\delta P_{i(+)}$, $\delta P^i_{ij(+)}$; for all $j \neq i$

- Calculate δF_i, $\delta P_{i(+)}$ and $\delta P^i_{ij(+)}$ from (14), (15) and (16) where in (14) $\delta P_{ij} = \delta P^i_{ij} + \delta P^{jT}_{ji}$, $\chi_{ij} = \chi^i_{ij} + \chi^{jT}_{ji}$, $\delta^2\chi_{ij} = \delta^2\chi^i_{ij} + \delta^2\chi^{jT}_{ji}$, and $(\cdot)^\dagger$ represents the pseudo-inverse which might be necessary because of the deadbeat property of discrete-time systems.

In *individual processing*, the global structure is very simple, since the amount of interactions between processors is minimal. In this type of processing, there is only one external phase of calculations. Separate phases of state covariance and feedback gain computations of the *team processing* are performed internally by each processor without the need of any communications. The global structure and the single phase of the *individual processing* is as follows:

Individual Processing: Global

1. Given the previous values of feedback gains through out the horizon, determine new values using the procedures in *Single Phase* described below.
2. Determine the trace of the cost function and if the difference between the current value of the cost and the value obtained from the previous iteration is small, the procedure ends.
3. Go back to step 1.

Individual Processing: Single Phase

Data Status
- Received: $\delta F_j^{(-)}(0), \ldots, \delta F_j^{(-)}(N-1)$; for all $j \neq i$
- Available: $\delta F_i^{(-)}(0), \ldots, \delta F_i^{(-)}(N-1)$
- Transmitted: $\delta F_i(0), \ldots, \delta F_i(N-1)$

1. Determine cross-covariance matrices $\delta\chi_{k_1 k_2}^{<i>}(1), \ldots, \delta\chi_{k_1 k_2}^{<i>}(N)$ and $\delta^2\chi_{k_1 k_2}^{<i>}(1), \ldots, \delta^2\chi_{k_1 k_2}^{<i>}(N)$; for all k_1 and k_2 from $\delta\chi_{k_1 k_2}^{<i>}(+) = [\delta\chi_{ik}(+)]_{\substack{i=k_1 \\ k=k_2}}$ and $\delta^2\chi_{k_1 k_2}^{<i>}(+) = \left[\delta^2\chi_{ij}^{i}(+) + \delta^2\chi_{ji}^{j^T}(+)\right]_{\substack{i=k_1,\, j=k_2 \\ \delta F=\delta F^{(-)}}}$ where $\delta\chi_{ik}(+)$ and $\delta^2\chi_{ij}^{i}(+)$ are given in (12) and (13), respectively. Here, $\delta\chi_{k_1 k_2}^{<i>}(0)$ and $\delta^2\chi_{k_1 k_2}^{<i>}(0)$ are zero.
2. If the difference between the new value of $\chi_{ii}^{<i>} + \rho[\delta\chi_{ii}^{<i>} + \delta\chi_{ii}^{<i>^T}] + \rho^2\delta^2\chi_{ii}^{<i>}$ and the previous value at the last time step is small enough, the procedure ends.
3. Calculate $\delta F_i(N-1)$, $\delta P_k^{<i>}(1)$, $\delta P_{k_1 k_2}^{<i>}(1), \ldots, \delta F_i(0)$, $\delta P_k^{<i>}(N)$, $\delta P_{k_1 k_2}^{<i>}(N)$, recursively for all k, $k_1 \neq k_2$ using $\delta P_k^{<i>} = \delta P_k$ in (15) and $\delta P_{k_1 k_2}^{<i>} = \delta P_{k_1 k_2}^{k_1} + \delta P_{k_2 k_1}^{k_2^T}$ in (16) and from (14) where in the equation $\delta P_k^{<i>}$, $\delta P_{k_1 k_2}^{<i>}$, $\delta\chi_{k_1 k_2}^{i}$ and $\delta^2\chi_{k_1 k_2}^{i}$ are used instead of δP_k, $\delta P_{k_1 k_2}$, $\delta\chi_{k_1 k_2}$ and $\delta^2\chi_{k_1 k_2}$, respectively. Here, $\delta P_k^{<i>}(0)$ and $\delta P_{k1k2}^{<i>}(0)$ are zero for all distinct subsystems k_1 and k_2.
4. Go back to step 1.

In the extended version of this work, we show these methods converge at least linearly, under the assumptions given in the previous section. The convergence is shown by showing an increase in the order of the errors between the optimal feedback gains and the feedback gains computed at each iteration as a function of ρ. Due to space limitations, these results are not included here.

In the next section, we consider an example to demonstrate the performance of the above methods.

IV. EXAMPLE

As an example, we consider an interconnected system given in [8]. The system consists of three interconnected two dimensional subsystems where the dynamics is described by

$$x(+) = \begin{bmatrix} 5.0 & 0.6 & \vdots & 0.0 & 0.0 & \vdots & 0.0 & 0.1 \\ 1.0 & -0.7 & \vdots & 0.0 & 0.0 & \vdots & 0.0 & 0.0 \\ \cdots & \cdots & \cdot & \cdots & \cdots & \cdot & \cdots & \cdots \\ 0.0 & 0.9 & \vdots & -0.6 & 0.0 & \vdots & 0.0 & 0.0 \\ 0.0 & 0.0 & \vdots & 1.0 & -0.4 & \vdots & 0.0 & 0.0 \\ \cdots & \cdots & \cdot & \cdots & \cdots & \cdot & \cdots & \cdots \\ 0.0 & 0.0 & \vdots & 0.8 & 0.0 & \vdots & -0.8 & 0.0 \\ 0.0 & 0.0 & \vdots & 0.0 & 0.0 & \vdots & 0.4 & -0.3 \end{bmatrix} x + \begin{bmatrix} 0.8 & \vdots & 0.0 & \vdots & 0.0 \\ 0.1 & \vdots & 0.0 & \vdots & 0.0 \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ 0.0 & \vdots & 1.2 & \vdots & 0.0 \\ 0.0 & \vdots & 0.0 & \vdots & 0.0 \\ \cdots & \cdot & \cdots & \cdot & \cdots \\ 0.0 & \vdots & 0.0 & \vdots & 1.5 \\ 0.0 & \vdots & 0.0 & \vdots & 0.0 \end{bmatrix} u \qquad (11)$$

The weighting matrices H_i, Q_i and R_i used in the decentralized cost (3) are assumed to be all identity matrices. For simulation purposes, we assumed $\rho = 0.9$, and the initial conditions are $x_i(+) = (0, 10)^T$, for all $i = 1, 2, 3$. We compared the performances of the proposed methods in Table 1. We included the *nominal* decentralized control, which does not consider interaction effects, and the centralized control in the Table. The multiple entries in the number of minimum and maximum iterations of the *individual processing* method correspond to the numbers of iterations in processors. For convergence, we assume the difference between the current and the previous state covariance matrices in the *single phase* of the *individual processing* is less than 10^{-3}, and the difference between the costs in the *global* phases of the methods is at least 10^{-6}. We also limit the number of iterations to 100.

Table 1: Performances of different distributed computations.

hori- zon	*decentr.; nominal*	*decentralized; individual processing*			*decentralized; team processing*		*central- ized*
length	cost	cost	# min iter. # max iter.	# global iterations	cost	# iter.	cost
1	244.92	244.92	1,1,1 1,1,1	2	244.92	2	220.82
5	673.57	486.76	1,1,1 5,14,10	15	486.76	22	335.29
10	758.79	503.53	4,17,5 22,100,68	100	503.67	100	352.95
20	765.26	514.09	5,8,1 100,100,100	100	507.29	100	355.03
30	765.28	505.63	1,1,1 5,62,22	100	507.29	100	355.05
40	765.28	505.82	1,1,1 2,44,4	100	506.09	100	355.03

As we observe from the Table in a range of horizon lengths the *team processing* method performs better[2] under limited number of iterations constraint. The *individual processing* method converges much slower partly because of the minimal information transfers between processors. As the horizon length increases, the convergence rate of both of these methods increases and the *individual processing* method becomes advantages because of its minimal use of outside information. We also observed that the limit of the number of iterations should be carefully chosen. For some horizon lengths, if this limit is small, the feedback gains will not even be near optimal.

---

[2] The difference in the costs of the two methods at horizon length 20 is due to the forced termination of the methods at 100 iterations and the different convergence rates of each methods.

The choice of these methods for an application depends on the communication costs and delays between processors and the horizon length. For very short and very long horizon lengths, the *individual processing* method may yield reasonable performance with minimal communications. On the other hand, for other horizon lengths the *team processing* method provides better performance for smaller number of iterations but with considerable information exchanges.

V. CONCLUSION

In this work, we have examined two different types of distributed processing strategies which iteratively solve the optimal decentralized feedback gains. The first strategy which we called the *team processing* uses most recent information from other processors and needs massive data transfers. The second strategy is called *individual processing* and this type minimizes data transfers among processors.

These two strategies have different places of application. The *team processing* is suitable, when a common memory can be shared by all the processors or when data transfer is easy and not time consuming. On the other hand, the *individual processing* method is suitable for parallel processors with isolated memory allocations and time consuming communication means.

We also observed the performances of these strategies in an example. Comparing only the total number of iterations versus costs, we conclude that the *team processing* method converges faster. On the other hand, if delays in the communication links and synchronization problems are included, the *individual processing* method may be preferable.

VI. REFERENCES

[1] M. Hodžić and D. D. Šiljak. Iterative methods for parallel-multirate estimation. In *Preprints of the 4th IFAC/IFORS Symposium*, pages 169–175, Aug. 1986.

[2] W. E. Moore. The parallel implementation of a cascade adaptive identification algorithm. *IEEE Transactions on Automatic Control*, AC-31:1074–1076, Nov. 1986.

[3] K. Malinowski. A. Y. Allidina, M. G. Singh, and W. D. Crorkin. Decomposition-coordination techniques for parallel simulation, part 1. *Large Scale Systems, Theory and Applications*, 9(2):101–115, Oct. 1985.

[4] L. Acar and Ü. Özgüner. A locally computed suboptimal control strategy for interconnected systems. (Submitted for publication).

[5] R. Bellman. *Dynamic Programming.* Princeton Univ., Princeton, NJ, 1957.

[6] R. Boudarel, J. Delmas, and P. Guichet. *Dynamic Programming and its Application to Optimal Control.* Academic Press, 1971.

[7] G. Chen and Jr. W. H. Mills. Finite elements and terminal penalization for quadratic cost optimal control problems governed by ordinary differential equations. *SIAM Journal of Control and Optimization*, 19(6), Nov. 1981.

[8] A. E. Yanchevsky and V. J. Hirkonen. Optimization of feedback systems with constrained information flow. *International Journal on Systems Sciences*, 12:1459–1468, 1981.

APPENDIX

In the appendix, we include some of the formulas which are referenced in the previous sections.

$$\begin{aligned}
\delta\chi_{ik(+)} &= (\tilde{A}_i + B_i\tilde{F}_i)\delta\chi_{ik}(\tilde{A}_k + B_k\tilde{F}_k)^T + B_i\delta F_i\chi_{ik}(\tilde{A}_k + B_k\tilde{F}_k)^T + \sum_{s\neq i}\delta A_{is}\chi_{sk}(\tilde{A}_k + B_k\tilde{F}_k)^T \\
&\quad + \rho\biggl\{ B_i\delta F_i\delta\chi_{ik}(\tilde{A}_k + B_k\tilde{F}_k)^T + \sum_{s\neq i}\delta A_{is}\delta\chi_{sk}(\tilde{A}_k + B_k\tilde{F}_k)^T \biggr\},
\end{aligned} \tag{12}$$

$$\begin{aligned}
\delta^2\chi^i_{ij(+)} &= (\tilde{A}_i + B_i\tilde{F}_i)\delta^2\chi^i_{ij}(\tilde{A}_j + B_j\tilde{F}_j)^T + B_i\delta F_i\chi^i_{ij}\delta F_j^T B_j^T + (\tilde{A}_i + B_i\tilde{F}_i)\delta\chi_{ij}\delta F_j^T B_j^T \\
&\quad + \sum_{t\neq j}\Bigl[(\tilde{A}_i + B_i\tilde{F}_i)\delta\chi_{it}\delta A_{jt}^T + B_i\delta F_i\chi^i_{it}\delta A_{jt}^T + \delta A_{it}\chi^t_{tj}\delta F_j^T B_j^T\Bigr] + \sum_{s\neq i}\sum_{t\neq j}\delta A_{is}\chi^s_{st}\delta A_{jt}^T \\
&\quad + \rho\biggl\{ B_i\delta F_i\delta\chi_{ij}\delta F_j^T B_j^T + (\tilde{A}_i + B_i\tilde{F}_i)\delta^2\chi^i_{ij}\delta F_j^T B_j^T + B_i\delta F_i\delta^2\chi^i_{ij}(\tilde{A}_j + B_j\tilde{F}_j)^T \\
&\qquad + \sum_{t\neq j}\bigl[B_i\delta F_i\delta\chi_{it}\delta A_{jt}^T + \delta A_{it}\delta\chi_{tj}\delta F_j^T B_j^T\bigr] + \sum_{t\neq j}\sum_{s\neq i}\delta A_{is}\delta\chi_{st}\delta A_{jt}^T \\
&\qquad + \sum_{t\neq i}\Bigl[(\tilde{A}_i + B_i\tilde{F}_i)\delta^2\chi^i_{it}\delta A_{jt}^T + \delta A_{it}\delta^2\chi^t_{tj}(\tilde{A}_j + B_j\tilde{F}_j)^T\Bigr]\biggr\} + \rho^2\biggl\{ B_i\delta F_i\delta^2\chi^i_{ij}\delta F_j^T B_j^T \\
&\qquad + \sum_{t\neq j}B_i\delta F_i\delta^2\chi^i_{it}\delta A_{jt}^T + \sum_{s\neq i}\delta A_{is}\delta^2\chi^s_{sj}\delta F_j^T B_j^T + \sum_{t\neq j}\sum_{s\neq i}\delta A_{is}\delta^2\chi^s_{st}\delta A_{jt}^T\biggr\},
\end{aligned} \tag{13}$$

$$\begin{aligned}
\delta F_i &= -[R_i + B_i^T(\tilde{P}_i + \rho\delta P_i)B_i]^{-1}B_i^T\biggl[\delta P_i(\tilde{A}_i + B_i\tilde{F}_i)\chi_{ii} + \sum_{j\neq i}[\delta P_{ij}(\tilde{A}_i + B_i\tilde{F}_i) + \tilde{P}_i\delta A_{ij}]\chi_{ji} \\
&\quad + \rho\biggl\{\delta P_i(\tilde{A}_i + B_i\tilde{F}_i)[\delta\chi_{ii} + \delta\chi_{ii}^T] + \sum_{j\neq i}\delta P_i\delta A_{ij}\chi_{ji} \\
&\qquad + \sum_{j\neq i}[\tilde{P}_i\delta A_{ij} + \delta P_{ij}(\tilde{A}_i + B_i\tilde{F}_i)][\delta\chi_{ji} + \delta\chi_{ij}^T] + \sum_{j\neq i}\sum_{s\neq j}\delta P_{ij}\delta A_{js}\chi_{si}\biggr\}
\end{aligned}$$

$$+\rho^2 \left\{ \delta P_i(\tilde{A}_i + B_i\tilde{F}_i)\delta^2\chi_{ii} + \sum_{j\neq i}\delta P_i\delta A_{ij}[\delta\chi_{ji} + \delta\chi_{ij}^T] + \sum_{j\neq i}\sum_{s\neq j}\delta P_{ij}\delta A_{js}[\delta\chi_{si} + \delta\chi_{is}^T] \right.$$

$$\left. + \sum_{j\neq i}[\tilde{P}_i\delta A_{ij} + \delta P_{ij}(\tilde{A}_i + B_i\tilde{F}_i)]\delta^2\chi_{ji} \right\} + \rho^3 \left\{ \sum_{j\neq i}\delta P_i\delta A_{ij}\delta^2\chi_{ji} \right.$$

$$\left. + \sum_{j\neq i}\sum_{s\neq j}\delta P_{ij}\delta A_{js}\delta^2\chi_{si} \right\} \Bigg] \left(\chi_{ii} + \rho[\delta\chi_{ii} + \delta\chi_{ii}^T] + \rho^2\delta^2\chi_{ii}\right)^{\dagger} , \tag{14}$$

$$\delta P_{i(+)} = (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_i(\tilde{A}_i + B_i\tilde{F}_i)$$

$$+ \left[R_i\tilde{F}_i + B_i^T\tilde{P}_i(\tilde{A}_i + B_i\tilde{F}_i)\right]^T \delta F_i + \delta F_i^T\left[R_i\tilde{F}_i + B_i^T\tilde{P}_i(\tilde{A}_i + B_i\tilde{F}_i)\right]$$

$$+\rho\left\{ \delta F_i^T(R_i + B_i^T\tilde{P}_iB_i)\delta F_i + (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_iB_i\delta F_i + \delta F_i^TB_i^T\delta P_i(\tilde{A}_i + B_i\tilde{F}_i) \right.$$

$$\left. + \sum_{j\neq i}\left[\delta A_{ji}^T\tilde{P}_j\delta A_{ji} + (\tilde{A}_i + B_i\tilde{F}_i)^T[\delta P_{ij}^i + \delta P_{ji}^{j^T}]\delta A_{ji} + \delta A_{ji}^T[\delta P_{ji}^j + \delta P_{ij}^{i^T}](\tilde{A}_i + B_i\tilde{F}_i)\right] \right\}$$

$$+\rho^2\left\{ \delta F_i^TB_i^T\delta P_iB_i\delta F_i + \sum_{j\neq i}\left[\delta F_i^TB_i^T[\delta P_{ij}^i + \delta P_{ji}^{j^T}]\delta A_{ji} + \delta A_{ji}^T[\delta P_{ji}^j + \delta P_{ij}^{i^T}]B_i\delta F_i \right.\right.$$

$$\left.\left. + \delta A_{ji}^T\delta P_j\delta A_{ji}\right] + \sum_{j_1\neq i}\sum_{j_2\neq i,j_1}\delta A_{j_1i}^T[\delta P_{j_1j_2}^{j_1} + \delta P_{j_2j_1}^{j_2^T}]\delta P_{j_1j_2}\delta A_{j_2i} \right\} , \tag{15}$$

$$\delta P_{ij(+)}^i = (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_{ij}^i(\tilde{A}_j + B_j\tilde{F}_j) + (\tilde{A}_i + B_i\tilde{F}_i)^T\tilde{P}_i\delta A_{ij} + \rho\left\{ (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_i\delta A_{ij} \right.$$

$$+ (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_{ij}^iB_j\delta F_j + \delta F_i^TB_i^T\delta P_{ij}^i(\tilde{A}_j + B_j\tilde{F}_j) + \delta F_i^TB_i^T\tilde{P}_i\delta A_{ij}$$

$$\left. + \sum_{s\neq i,j}\left[\frac{1}{2}\delta A_{si}^T\tilde{P}_s\delta A_{sj} + (\tilde{A}_i + B_i\tilde{F}_i)^T\delta P_{is}^i\delta A_{sj} + \delta A_{si}^T\delta P_{sj}^s(\tilde{A}_j + B_j\tilde{F}_j)\right] \right\}$$

$$+\rho^2\left\{ \delta F_i^TB_i^T\delta P_{ij}^iB_j\delta F_j + \delta F_i^TB_i^T\delta P_i\delta A_{ij} + \sum_{s\neq i,j}\left[\delta F_i^TB_i^T\delta P_{is}^i\delta A_{sj} + \delta A_{si}^T\delta P_{sj}^sB_j\delta F_j \right.\right.$$

$$\left.\left. + \frac{1}{2}\delta A_{si}^T\delta P_s\delta A_{sj}\right] + \sum_{s\neq i}\sum_{t\neq j,s}\delta A_{si}^T\delta P_{st}^s\delta A_{ti} \right\} . \tag{16}$$

A STABILITY THEORY OF LINEAR TIME VARYING SYSTEMS

PAUL R. BOUTHELLIER AND BIJOY K. GHOSH
Department of Systems Science & Mathematics
Washington University
St. Louis, Missouri 63130

ABSTRACT

In this paper we obtain stability results for a certain class of linear time varying systems. Our interest in the above stems from a desire to develop a robust stabilization theory via an adaptive control. The stability results enable us to consider compensators for a structured family of time varying systems.

I. INTRODUCTION

In this paper we shall consider the Bounded input-bounded output stability properties of the class of single input-single output linear time varying ARMA systems of order n, i.e.

$$y_k - \sum_{i=1}^{n} \alpha_{i,k} y_{k-i} = \sum_{i=1}^{n} \beta_{i,k} u_{k-i} \tag{1.1}$$

The extensive use of models of the form (1.1) in mathematics, engineering and economics necessitates development of a stability theory which incorporates possible variations of the parameters $\alpha_{i,k}$ and $\beta_{i,k}$ $i=1,\ldots,n$. Such variations may originate via estimation errors, updating schemes or a change in the perceived importance of past data.

In Section II we develop a theory of stabilizing ARMA systems of the form (1.1) for the case where $\alpha_{i,k}$ and $\beta_{i,k}$ converge to fixed values as $k\to\infty$. It will be shown that the stability properties of such systems are the same as those they converge to.

In Section III the notion of a generalized stability region is defined. The results of Section II will enable us to prove that there exists a generalized stability region about each stable system of the form (1.1). This will then allow us to construct families of stable time varying systems. Section IV will conclude with a brief indication for future research.

II. STABILITY FOR A CLASS OF LTV PLANTS

We shall consider the stability properties of a single input-single output LTV ARMA system

$$y_k - \sum_{i=1}^{n} \alpha_{i,k} y_{k-i} = \sum_{i=1}^{n} \beta_{i,k} u_{k-i} \tag{2.1}$$

where the corresponding parameter vector $(\alpha_{1,k},\ldots,\alpha_{n,k};\beta_{1,k},\ldots,\beta_{n,k})$ converges as $k\to\infty$ to a fixed point in $\mathbb{R}^{2n}$. Such a problem arises in i) parameter estimation problems and ii) robust stabilization problems [4]. The main result of this section is now given.

Theorem 2.1: Consider the LTV system

$$y_k - \alpha_{1,k}y_{k-1} - \cdots - \alpha_{n,k}y_{k-n} = \beta_{1,k}u_{k-1} + \cdots + \beta_{n,k}u_{k-n}$$

where $\alpha_{i,k} \xrightarrow[k\to\infty]{} \alpha_i \in (-\infty,\infty)$ and $\beta_{i,k} \xrightarrow[k\to\infty]{} \beta_i \in (-\infty,\infty)$ $i=1,\ldots,n$. The LTV system is stable iff the LTI system it converges to is stable.

Proof: The proof will consist of three parts. Part I will consider convergence to a stable system with distinct eigenvalues. Part II will consider convergence to a stable system which possibly has multiple eigenvalues. Part III, the unstable case, is rather tedious and hence omitted. We refer the interested reader to [1]. It will be noted that while the proof in Part II is a generalization of the proof in Part I, the proof constructed in Part I will yield an interesting result which will be discussed in Section III.

For sake of notation we define

$$J(\lambda,k) \triangleq \begin{bmatrix} \lambda & k & & 0 \\ & \ddots & \ddots & \\ & & \ddots & k \\ 0 & & & \lambda \end{bmatrix}$$

where the dimension is understood.

Part I: Consider the case of

$$y_k - \alpha_1 y_{k-1} - \cdots - \alpha_n y_{k-n} = \beta_1 u_{k-1} + \cdots + \beta_n u_{k-n}$$

being a stable system and $\lambda^n - \alpha_1\lambda^{n-1} - \cdots - \alpha_{n-1}\lambda - \alpha_n = 0$ having distinct roots. We first write

$$y_k - \alpha_{1,k}y_{k-1} - \cdots - \alpha_{n,k}y_{k-n} = \beta_{1,k}u_{k-1} + \cdots + \beta_{n,k}u_{k-n}$$

in "state-space" form [5].

Define

i) $x_i(k) \triangleq y_{k+(i-n)}$ $i=1,2,\ldots,n$

ii) $\tilde{u}_k \triangleq \beta_{1,k}u_k + \beta_{2,k}u_{k-1} + \cdots + \beta_{n,k}u_{k-n+1}$

It is easy to show that

$$\underset{\sim}{x}_{k+1} = A_k \underset{\sim}{x}_k + B\tilde{u}_k, \qquad y_k = C\underset{\sim}{x}_k \tag{2.2}$$

where $\underset{\sim}{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}$; $A_k = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & \\ 0 & 0 & \cdots & \cdots & 0 & 1 \\ \alpha_{n,k+1} & \cdots & & \cdots & \alpha_{2,k+1} & \alpha_{1,k+1} \end{bmatrix}$; $B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$

$C = [0 \ldots 0\ 1]$

and $\underset{\sim}{x}(0) = \underset{\sim}{0}$ (As it is assumed that $y_i = 0 \quad i \le 0$)

Define the state transition matrix for (2.1) by

$$\Omega(i,j) = \begin{cases} A(i-1)A(i-2) \ldots A(j), & i>j, \\ I, & i=j, \\ \text{not defined}, & i<j \end{cases}$$

A simple calculation yields

$$y_k = C \sum_{i=0}^{k-1} \Omega(k,i+1)B\tilde{u}_i$$

Assume the u_i are bounded. As $\{\beta_{i,k}\}_{k=1}^{\infty}$ is bounded $i=1,\ldots,n$ there exists $u^* < \infty$ such that $|\tilde{u}_i| \le u^*$ for all i.

Let $r_1, \ldots, r_n$ be the roots of $\lambda^n - \alpha_1\lambda^{n-1} - \cdots - \alpha_n = 0$. Define the corresponding roots of $\lambda^n - \alpha_{1,k}\lambda^{n-1} - \cdots - \alpha_{n-1,k}\lambda - \alpha_{n,k} = 0$ to be $r_1+\delta_{1,k}, \ldots, r_n+\delta_{n,k}$. As $\alpha_{i,k} \xrightarrow[k\to\infty]{} \alpha_i$ $i=1,\ldots,n$ for any $\varepsilon>0$ there exists $N_\varepsilon<\infty$ such that $\delta_{i,k}\in(-\varepsilon,\varepsilon)$ $i=1,\ldots,n$ $k\ge N_\varepsilon$.

Choose $\varepsilon_1>0$ such that

$$\text{i) } r_i+\delta_{i,k} \ne r_j+\delta_{j,k} \quad 1 \le i \ne j \le n$$

$$\text{ii) } |r_i+\delta_{i,k}| < \gamma < 1 \quad i = 1,\ldots,n$$

for all $k \ge N_{\varepsilon_1}$. As we are assuming the LTI system is stable and has distinct eigenvalues there exists such an ε_1. Let $k \ge N_{\varepsilon_1}$ and define $\lambda_{i,k} = r_i + \delta_{i,k}$ $i=1,\ldots,n$ to be the eigenvalues of A_k. Define the corresponding eigenvectors to be $v_{i,k}$ $i=1,\ldots,n$ respectively. It is easy to check that

$$v_{i,k} = [1,(r_i+\delta_{i,k}),\ldots,(r_i+\delta_{i,k})^{n-1}]^T$$

Define $T_k \overset{\Delta}{=} \left(\text{col}\left[v_{1,k}^T,\ldots,v_{n,k}^T\right]\right)^T$

$$T_k^{-1}A_kT_k = \Lambda_k \overset{\Delta}{=} \text{diag}[r_1+\delta_{1,k},\ldots,r_n+\delta_{n,k}]$$

implies $A_k = T_k\Lambda_kT_k^{-1}$.

Consider $\Omega(k+2,k) = T_{k+1}\Lambda_{k+1}T_{k+1}^{-1}T_k\Lambda_kT_k^{-1}$. By continuity, given any $\delta>0$ there exists $\varepsilon_\delta>0$ such that for all $k\ge N_{\varepsilon_\delta}$ $T_{k+1}^{-1}T_k = [t_{ij}]$ where

$$\begin{cases} |t_{ij}| < 1+\delta \ , & i=j \\ & \qquad 1\le i,j\le n \\ |t_{ij}| < \delta \ , & i\ne j \end{cases}$$

Denote this by $T_{k+1}^{-1}T_k <^* \begin{bmatrix} 1+\delta & \delta & \delta & \dots & \delta \\ \delta & 1+\delta & \delta & \dots & \delta \\ & & \vdots & & \\ \delta & \delta & \dots & \delta & 1+\delta \end{bmatrix} \triangleq \bar{\Gamma}$.

Choose δ such that $\gamma(1+n\delta) < 1$ and let $k \ge \max\{N_{\varepsilon_1}, N_{\varepsilon_\delta}\}$. Therefore

$$\Lambda_{k+1}T_{k+1}^{-1}T_k\Lambda_k <^* \begin{bmatrix} \gamma & & & \bigcirc \\ & \gamma & & \\ & & \ddots & \\ \bigcirc & & & \gamma \end{bmatrix} \begin{bmatrix} 1+\delta & \delta & \dots & \delta \\ \delta & 1+\delta & \dots & \delta \\ & \vdots & & \\ \delta & \delta & \dots \ \delta & 1+\delta \end{bmatrix} \begin{bmatrix} \gamma & & & \bigcirc \\ & \gamma & & \\ & & \ddots & \\ \bigcirc & & & \gamma \end{bmatrix}$$

$$= \gamma^2 \begin{bmatrix} 1+\delta & \delta & \dots & \delta \\ \delta & 1+\delta & \dots & \delta \\ & \vdots & & \\ \delta & \delta & \dots \ \delta & 1+\delta \end{bmatrix} . \tag{2.3}$$

Consider $\Omega(k+n+1,k) = \prod_{i=0}^{n} T_{k+i}\Lambda_{k+i}T_{k+i}^{-1}$.

As $T_k \xrightarrow[k\to\infty]{} T = (\text{col}\begin{bmatrix} v_1^T, \dots, v_n^T \end{bmatrix})^T$ where $v_i = \begin{bmatrix} 1, \lambda_i, \dots, \lambda_i^{n-1} \end{bmatrix}^T$, T_k and T_k^{-1} are bounded in the sense of $<^*$ for all $k\ge N$. Let L_1 and L_2 be their upper bounds respectively, defined by $L_1 \triangleq [m_1]$ where $m_1 = \max_{i,j}|(T_k)_{i,j}|$ and $L_2 \triangleq [m_2]$ where $m_2 = \max_{i,j}|(T_k^{-1})_{i,j}|$.

By an obvious extension of (2.3)

$$\Lambda_{k+n}T_{k+n}^{-1}T_{k+n-1}\Lambda_{k+n-1} \cdots \Lambda_{k+1}T_{k+1}^{-1}T_k\Lambda_k <^* \gamma^{n+1}\bar{\Gamma}^n .$$

The eigenvalues of $\bar{\Gamma}$ are $1+n\delta$ and 1, 1 repeated $n-1$ times.

Define $$\Gamma \triangleq \begin{bmatrix} \gamma(1+\delta) & \gamma\delta & \dots & \gamma\delta \\ \gamma\delta & \gamma(1+\delta) & \dots & \gamma\delta \\ & \vdots & & \\ \gamma\delta & \gamma\delta & \dots & \gamma(1+\delta) \end{bmatrix}$$

Therefore $\Lambda_{k+n}T^{-1}_{k+n}T_{k+n-1}\Lambda_{k+n-1}\cdots\Lambda_{k+1}T^{-1}_{k+1}T_{k}\Lambda_{k} \overset{*}{<} \gamma^{n+1-n}\Gamma^{n} = \gamma\Gamma^{n}$. Γ has eigenvalues γ and $\gamma(1+n\delta)$, γ repeated $n-1$ times. Let Q be such that

$$Q^{-1}\Gamma Q = \mathrm{diag}[\gamma(1+n\delta), J(\gamma,1),\ldots,J(\gamma,1)]$$

Define $J \overset{\Delta}{=} Q^{-1}\Gamma Q$

As $\gamma<1$ and $\gamma(1+n\delta)<1$ $J^{n}\underset{n\to\infty}{\to}[0]$. Thus $\gamma\Gamma^{n} = \gamma QJ^{n}Q^{-1}\underset{n\to\infty}{\to}[0]$. Hence

$$\Omega(\infty,j) \overset{\Delta}{=} (\lim_{m\to\infty}\Omega(m+1,n+1))\Omega(n+1,j) = [0] \quad j=2,\ldots,n-1. \tag{2.4}$$

Assume $k>N$.

$$y_{k} = C\sum_{i=0}^{k-1}\Omega(k,i+1)B\tilde{u}_{i} = C\sum_{i=0}^{N}\Omega(k,i+1)B\tilde{u}_{i}+C\sum_{i=N}^{k}\Omega(k,i+1)B\tilde{u}_{i}$$

i) $C\sum_{i=0}^{N-1}\Omega(k,i+1)B\tilde{u}_{i}\underset{k\to\infty}{\to}0$ by (2.4), and

ii) $C\sum_{i=N}^{k-1}\Omega(k,i+1)B\tilde{u}_{i} \overset{*}{<} C\sum_{i=N+1}^{k}L_{1}\gamma\Gamma^{k-i-1}L_{2}Bu^{*} = \gamma CL_{1}Q\sum_{i=N+1}^{k}J^{k-i-1}Q^{-1}u^{*}$

$\overset{*}{<}\gamma CL_{1}Q\sum_{i=1}^{\infty}J^{i}Q^{-1}u^{*}$ for all $k>N$. It is easy to show that $\sum_{i=1}^{\infty}J^{i}$ is bounded, hence, via i) and ii) we have our result.

Part II: All notation as in Part I

Define $$A \overset{\Delta}{=} \begin{bmatrix} 0 & 1 & 0 & \cdots & & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & \\ 0 & 0 & \cdots & & 0 & 1 \\ \alpha_{n} & \alpha_{n-1} & \cdots & & \alpha_{2} & \alpha_{1} \end{bmatrix}$$

Let λ_{i} $i=1,\ldots,\alpha$ be the non-repeated eigenvalues of A and λ_{i} $i=\alpha+1,\ldots,\beta$ the repeated eigenvalues. There exists a matrix M such that

$$M^{-1}AM = \mathrm{diag}[\lambda_{1},\ldots,\lambda_{\alpha},J(\lambda_{\alpha+1},1),\ldots,J(\lambda_{\beta},1)]$$

Define M_{k} to be the nxn matrix whose i^{th} column is $k^{i-1}\underset{\sim}{e}_{i}$ where $\underset{\sim}{e}_{1},\ldots,\underset{\sim}{e}_{n}$ are the columns of M. It can be shown that

$$M_{k}^{-1}AM_{k} = \mathrm{diag}[\lambda_{1},\ldots,\lambda_{\alpha},J(\lambda_{\alpha+1},k),\ldots,J(\lambda_{\beta},k)]$$

Define $\lambda \overset{\Delta}{=} \max_{1\le i\le\beta}|\lambda_{i}|$. Choose $\delta>0$ such that $\lambda+n\delta<\gamma<1$. By continuity and the fact that k can be choosen arbitrarily small, there exists $N<\infty$ such that for all $n>N$

$$M_k^{-1} A_n M_k <^* \begin{bmatrix} \lambda+\delta & \delta & \cdot\ \cdot, \cdot & \delta \\ \delta & \lambda+\delta & \cdot\ \cdot\ \cdot & \delta \\ & & \vdots & \\ \delta & \delta & \cdot\ \cdot\ \cdot\ \lambda+\delta & \delta \\ \delta & \delta & \cdot\ \cdot\ \cdot\ \delta & \lambda+\delta \end{bmatrix} \overset{\Delta}{=} H .$$

Therefore $\prod_{i=N+1}^{n} M_k^{-1} A_i M_k = M_k^{-1} \Omega(n+1,N+1)\, M_k <^* H^{n-N}$. As the eigenvalues of H are $\lambda+n\delta$ and λ, λ repeated $n-1$ times, there exists a matrix Q such that

$$Q^{-1} H Q = \operatorname{diag}[\lambda+n\delta, J(\lambda,1), \ldots, J(\lambda,1)]$$

Thus $M_k^{-1}\Omega(n+1,N+1)M_k <^* QJ^{n-N}Q^{-1}$. As $\lim_{n\to\infty} J^{n-N} = [0]$ and $\sum_{i=1}^{\infty} J^i$ is bounded

i) $\lim_{n\to\infty} \Omega(n+1,N+1) = [0]$

ii) $\sum_{i=N}^{n-1} \Omega(n+1,i)$ is bounded for all $n>N$.

Assume $n>N$.

As before $y_n = C \sum_{i=0}^{n-1} \Omega(n,i+1)B\tilde{u}_i = C \sum_{i=0}^{N} \Omega(n,i+1)B\tilde{u}_i + C \sum_{i=N}^{n} \Omega(n,i+1)B\tilde{u}_i$.

The result follows from i) and ii) above. Q.E.D.

III. A GENERALIZED STABILITY REGION

In an obvious way Theorem 2.1 can be used to show that about every point $p \varepsilon \mathrm{IR}^n$ corresponding to a stable system, there exists a 'generalized stability region'. The use of this result in stability theory is clear. We will then show how these regions may be utilized to allow the coefficients of our time varying ARMA system to move through the parameter space while maintaining Bounded input-Bounded output stability.

Definition: A generalized stability region (g.s.r) about a point $p \in \mathrm{IR}^{2n}$ is an open neighborhood $U_p \subset \mathrm{IR}^{2n}$ about p with the property that if $\{(\alpha_{1,k},\alpha_{2,k},\ldots,\alpha_{n,k};\ \beta_{1,k},\ldots,\beta_{n,k})_{k=N}^{\infty} \in U_p$ for some $N<\infty$ the system

$$y_k - \alpha_{1,k} y_{k-1} \ldots - \alpha_{n,k} y_{k-n} = \beta_{1,k} u_{k-1} + \ldots + \beta_{n,k} u_{k-n}$$

is stable.

Define $S_n \overset{\Delta}{=} \{(\alpha_1,\ldots,\alpha_n;\ \beta_1,\ldots,\beta_n) \in \mathrm{IR}^{2n} \mid y_k - \alpha_1 y_{k-1} - \ldots - \alpha_n y_{k-n} = \beta_1 u_{k-1} + \ldots + \beta_n u_{k-n}$ is stable$\}$.

Theorem 3.1: About each point $p\in S_n$ there exists a g.s.r. U_p.

Proof: The proof is in two parts. Part I will prove the result for $p\in S_n$ corresponding to systems with distinct eigenvalues. This proof will allow us to construct a family of stable LTV systems. The proof of Part II, considering multiple eigenvalues, is similar to that of Part I and hence omitted. We refer the interested reader to [1].

In Part I the notation of Theorem 2.1 will be used.

Part I: Let $(\alpha_1,\dots,\alpha_n;\ \beta_1,\dots,\beta_n)\in S_n$ correspond to a system with distinct eigenvalues $\lambda_1,\dots,\lambda_n$. $T^{-1}AT=\Lambda=\begin{bmatrix}\lambda_1 & & 0\\ & \ddots & \\ 0 & & \lambda_n\end{bmatrix}$. By continuity, a region $\bar{F}_1 \overset{\Delta}{=} \mathop{X}_{i=1}^{n} (\alpha_i-\varepsilon_i,\alpha_i+\varepsilon_i)$ $\varepsilon_i>0$ $i=1,\dots,n$ sufficiently small, can be constructed satisfying

i) For each $\bar{\alpha}=(\bar{\alpha}_1,\dots,\bar{\alpha}_n)\in\bar{F}_1$, $\lambda^n-\bar{\alpha}_1\lambda^{n-1}-\cdots-\bar{\alpha}_n=0$ has distinct roots $\lambda_{1,\bar{\alpha}},\dots,\lambda_{n,\bar{\alpha}}$ which satisfy $|\lambda_{i,\bar{\alpha}}|<\gamma<1$ and

ii) if $\alpha,\alpha^*\in\bar{F}_1$ then

$$T_\alpha^{-1}T_{\alpha^*}<^*\begin{bmatrix}1+\delta & \delta & \dots & \delta\\ \delta & 1+\delta & \dots & \delta\\ & \vdots & & \\ \delta & \delta & \dots\ \delta & 1+\delta\end{bmatrix}$$

where $\gamma(1+n\delta)<1$. It is easy to check, following the steps of the proof of Theorem 2.1 Part I that $\bar{F}_1\times\bar{F}_2$, where $\bar{F}_2\overset{\Delta}{=}\mathop{X}_{i=1}^{n}(\beta_i-\Delta_i,\beta_i+\Delta_i)$ $0<\Delta_i<\infty$ $i=1,\dots,n$, is a generalized stability region for $(\alpha_1,\dots,\alpha_n;\ \beta_1,\dots,\beta_n)$

Q.E.D.

Note: Unless otherwise stated the generalized stability region of a point $p\in S_n$ is taken to be $U_p^*\overset{\Delta}{=}U_p\cap(\mathbb{R}^n\times\underset{\sim}{0}^n)$, the coefficients of the right hand side are assumed bounded. I.e., if $U_p=\mathop{X}_{i=1}^{n}(\alpha_i-\varepsilon_i,\alpha_i+\varepsilon_i)\times\mathop{X}_{i=1}^{n}(\beta_i-\Delta_i,\beta_i+\Delta_i)$ then $U_p^*=\mathop{X}_{i=1}^{n}(\alpha_i-\varepsilon_i,\alpha_i+\varepsilon_i)$. $*$ will be omitted where the meaning is clear.

Let us note what Theorem 3.1 says. About any point $p\in S_n$ there exists an open neighborhood U_p about p with property that any n-th order ARMA system evolving on U_p whose r.h.s. coefficients are bounded is stable. Let us now extend this result.

Define $S_n(\gamma,\beta) \overset{\Delta}{=} \{\underset{\sim}{\theta}\in S_n \mid$ i) $|\lambda|<\gamma<1$ where $\lambda = \max_{1\le i\le n}|\lambda_i|$ and $\lambda_i^n-\alpha_1\lambda_i^{n-1} - \cdots - \alpha_n\lambda_i-\alpha_n = 0$, ii) $\lambda_i \neq \lambda_j$ $i\neq j$, iii) $|\beta_i|<\beta<\infty$ $i=1,\ldots,n\}$.

About each point $p\in S_n(\gamma,\beta)$ there exists, by Theorem 3.1, a g.s.r. U_p about p which satisfies:

i) $|\lambda_{\max}|<\gamma$ where $\lambda_{\max} = \max_{1\le i\le n}\{\lambda_i\colon \lambda_i^n - \alpha_1\lambda_i^{n-1} - \cdots - \alpha_n = 0\}$ for all $(\alpha_1,\ldots,\alpha_n)\in U_p$

ii) $\lambda^n-\alpha_1\lambda^{n-1} - \cdots - \alpha_{n-1}\lambda-\alpha_n = 0$ has distinct roots for all $(\alpha_1,\ldots,\alpha_n)\in U_p$

iii)

$$T_{\underset{\sim}{\alpha}}^{-1}T_{\underset{\sim}{\beta}} \overset{*}{<} \begin{bmatrix} 1+\delta & \delta & \cdots & \delta \\ \delta & 1+\delta & \cdots & \delta \\ & & \vdots & \\ \delta & \delta & \cdots & 1+\delta \end{bmatrix} \quad \text{for all } \underset{\sim}{\alpha},\ \underset{\sim}{\beta} \in U_p$$

iv) $\gamma(1+n\delta) < 1$

v) $T_{\underset{\sim}{\alpha}}^{-1} \overset{*}{<} M < \infty$ for all $\underset{\sim}{\alpha} \in U_p$

where γ,δ and M are fixed.

Consider a sequence $\{U_{p_i}\}_{i=1}^{\infty}$ of g.s.r.'s corresponding to $\{p_i\}_{i=1}^{\infty}\in U_p$ which satisfy i)-v) above. Assume $U_{p_i} \cap U_{p_{i+1}} \neq \phi$ $i=1,2,\ldots$. Then the LTV system

$$y_k - \alpha_{1,k}y_{k-1} - \cdots - \alpha_{n,k}y_{k-n} = \beta_{1,k}u_{k-1} + \cdots + \beta_{n,k}u_{k-n}$$

satisfying

i) $(\alpha_{1,1},\alpha_{1,2},\ldots,\alpha_{n,1},\ \beta_{1,k},\ldots,\beta_{n,1}) \in U_{p_1}$

ii) $(\alpha_{1,k},\ldots,\alpha_{n,k};\ \beta_{1,k},\ldots,\beta_{n,k}) \in U_{p_{k-1}} \cap U_{p_k}$ $k=2,3,\ldots$

is stable. We will now formalize this into a theorem.

Theorem 3.2: Let $\{U_i\}_{i=1}^{\infty}$ be a sequence of g.s.r.'s satisfying

i) $U_i\in S_n(\gamma,\beta)$ $i=1,2,\ldots$ for some γ and β where $0<\gamma<1$ and $0<\beta<\infty$.

ii) $U_i \cap U_{i+1}\neq\phi$ $i = 1,2,\ \ldots$

iii) Each U_i satisfies (i-v) above.

Let $p_1 \in U_1$ and $p_1 \in U_i \cap U_{i+1}$ $i=1,2,\ldots$ be arbitrary. Then the siso LTV ARMA system whose parameter at time i is given by p_i is stable.

Proof: This follows immediately from the proof of Theorem 2.1 Part I.

IV. CONCLUSION

We have seen that about each point $p \varepsilon \mathrm{IR}^n$ corresponding to a stable system there exists a generalized stability region. These regions can be used to derive necessary and sufficient conditions for the existence of a non-switching compensator which stabilizes a linear time varying system [1]. Thus we may extend stabilization results derived for linear time invarient systems to the linear time varying case [2], [3], [6], [7].

REFERENCES

[1] P.R. Bouthellier, Ph.D. Dissertation, Washington University in St. Louis, in preparation.

[2] B.K. Ghosh, "Simultaneous Partial Pole Placement: A New Approach to Multimode System Design," IEEE Trans. Automat. Contr., vol. AC-31, pp. 440-443.

[3] B.K. Ghosh and C.I. Byrnes, "Simultaneous Stabilization and Simultaneous Pole Placement by Non-Switching Dynamic Compensation," IEEE Trans. Automat. Contr., vol. AC-28, pp. 735-741, 1983.

[4] G.C. Goodwin and K.S. Sin, "Adaptive Filtering, Prediction and Control," Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[5] T. Kailath, "Linear Systems," Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[6] M. Vidyasagar, "Control Systems Synthesis: A Factorization Approach," MIT Press, Cambridge, MA, 1985.

[7] D. Youla, J. Bongiorno, and C. Lu, "Single Loop Feedback Stabilization for Reliable Stabilization," Automatica, vol. 10, pp. 155-173, 1974.

FEEDBACK SYSTEMS WITH TAPERED AND OTHER SPECIAL INPUT SPACES

WILLIAM L. ROOT and DEMETRIOS SERAKOS
Aerospace Engineering Department
The University of Michigan
Ann Arbor, Michigan 48109-2140

ABSTRACT

In this paper the use of weighted input and output spaces in the mathematical description of feedback systems is examined. For a weighted input space, the recent past of an input signal is emphasized and the remote past is de-emphasized. Tapered spaces (sometimes referred to as fading memory spaces), are a specific subclass of weighted spaces. The use of weighted input and output spaces may produce results as regards the existence of a causal system and the properties of boundedness, continuity and stability that are different from what obtain with conventional input and output spaces. The results given here, largely for feedback systems, give some conditions for a system to be well-defined, bounded, continuous or stable. There is some difficulty in analyzing input-output systems with the time axis being the entire real line. It is sometimes avoided by using only the half line. A new device for easing this difficulty is presented. Some results concerning tapered input and output spaces used on linear systems are included.

INTRODUCTION

The problems considered in this paper have to do with analyzing the effect of weighted spaces, in particular tapered spaces, on various properties of a feedback system. The properties considered are the boundedness, causality, existence and the stability of solutions to the feedback equations. In the analysis of what may be called ''natural systems'' for example, the inputs to these systems may be measured but only over a finite time span, while the system will have been ''running'' for a long time previous to when the measurements are taken. It is thought that a taperedness assumption is reasonable for natural systems and necessary for their analysis. In another vein, Boyd and Chua [2] show that if a tapered input space is used approximations by Volterra series operators of nonlinear operators can be made over an infinite time interval.

The input and output spaces are metrized using an approach called the ''fitted families of seminorms'', developed by the first author, see [6] and [7]. This approach is more general (inclusive) than the procedure where time-projection operators are used on the spaces of time functions. Weighted (or tapered) spaces of time functions are more easily handled by the fitted families approach than by time-projections.

In the first part of this paper it is shown that some known general nonlinear feedback theorems, such as the small gain theorem, can be generalized to hold for weighted input and output spaces. Chapter III of [4] is referred to and three major results from it are presented using the more general fitted families approach to include tapered input and output spaces. In the second part of this paper results which specifically guarantee the causality of a feedback system are presented. These results have weaker hypotheses than results which guarantee causality, existence and boundedness, such as the incremental form of the small gain theorem. Results of this nature have not been obtained previously and they are presented here in the context of the fitted families. The results of these two sections are given for the case the time axis is the entire real line. Most researchers in input-output systems theory use the half line $[0,\infty)$ as the time axis, see [4] and [9], which may not always be desirable. A device is presented, a so-called *zero-shift* input space, to ease the difficulty of treating certain problems that essentially require formulation on the half line.

In the last section we consider linear, time-invariant systems. Necessary and sufficient conditions for BIBO stability of linear time invariant systems with exponentially tapered input and output spaces are given. The bounding space stability of a linear system with weighted input and output spaces is analyzed. These results can be applied to feedback systems.

PRELIMINARIES

An input-output system (Y,F,U) is, in the context of this paper, a mapping F from an input space U to an output space Y $(y=F(u))$. U is a translation invariant subset of a normed linear space A or an extended space A^e of time functions and similarly Y is a subset of the space B or B^e. A proper definition of the space A depends on the introduction of fitted families of seminorms and the corresponding normed linear spaces; this is done in [6] and [7], and A^e is defined in [7]. Suffice it to say here that an expanded fitted family of seminorms $\mathrm{N}=\{\|\cdot\|_{s,t};\ -\infty\le s\le t\le\infty,\ -\infty<t,\ s<\infty\}$ on a linear space $L=L(\mathbb{R},E)$ of time functions taking values in a Banach space E, is a translation invariant family of seminorms, arbitrary as long as some mild "fitting" conditions are satisfied, with the property that $\|f\|_{s,t}$ is determined only by the behavior of f on $[s,t]$ (or a subinterval thereof.) Associated with these seminorms are normed linear spaces $A_{s,t}$ where elements are equivalence classes of functions in L (for outputs we use $|||\cdot|||_{s,t}$ and $B_{s,t}$.) An example is the weighted L_p fitted family of normed linear spaces which are so defined: Let p be fixed, $1\le p\le\infty$ and let L be the set of strongly measurable (Lebesgue) functions f such that

$$\|f\|_{s,t}\overset{\Delta}{=}\left[\int_s^t\|f(\tau)\|^p\,w(t-\tau)\,d\tau\right]^{1/p}\tag{1}$$

exists for all $-\infty\le s\le t\le\infty$; $-\infty<t,\ s<\infty$. $(\|f\|_{s,\infty}\overset{\Delta}{=}\sup_{t\ge s}\|f\|_{s,t}.)$ The function $w:\mathbb{R}\to\mathbb{R}^+$ is the *weighting function* and has the following properties: (i) $w(0)=1$, (ii) is a monotone nonincreasing function. Equation (1) gives the standard L_p norm when $w(t)=1$ for all t. This will be referred to as the "flat" case. We define a particular type of a fitted family on a normed linear space.

Definition 1: [7] Consider a left expanded fitted family of normed linear spaces $\{A_{s,t}\}$; $-\infty\le s\le t<\infty$, $-\infty<t$. Define a subset $G(c,t)\subset A_t$ by

$$G(c,t)\overset{\Delta}{=}\{f_t\in A_t\ \mid\ \|f\|_s\le c\ \text{ for all } s\le t\}\ .$$

Then $\{A_{s,t}\}$ is a *tapered fitted family of normed linear spaces* if given $t\in\mathbb{R}$, $c>0$ and $\varepsilon>0$ there exists $\delta=\delta(\varepsilon,c,t)>0$ such that

$$\|f\|_s\le\|f\|_{s-\delta,s}+\varepsilon$$

for all $f\in G(c,t)$ and for all $s\le t$.

Note that because of the uniformity condition on G with respect to t, δ is actually independent of t, so that $\delta=\delta(\varepsilon,c)$. The weighted fitted family of normed linear spaces defined by (1) is tapered if w satisfies these additional properties: (iii) $w(t)\to0$ as $t\to\infty$, (iv) $\int_0 w(t)dt=W<\infty$.

A special type of time function will be useful:

Definition 2: Let $L=L(\mathbb{R},E)$ be a space of time functions with the property that for each $f\in L$ there is a real number $T=T(f)$ such that $f(t)=0$ for all $t<T$. Then L is called a *zero-shift space.* If a fitted family structure is defined on L, the corresponding bounding space A and extended space A^e are also called zero shift spaces.

The zero-shift assumption is an alternative to using the half line as the time axis. Not all systems have an on-off switch, so assuming the input space is a zero shift space is thought to be better than saying that the system always goes on at $t=0$.

Consider the following type of input space: assume that for all $u\in U$ (or U^e), $t\in\mathbb{R}$, $\|u\|_t=0$ implies $\|u\|_s=0$ for all $s\le t$. This assumption is satisfied by the fitted family of seminorms defined by (1) if $w(t)>0$ for all $t\in\mathbb{R}^+$. An interpretation is that the remote past of a time function may be de-emphasized in computing its norm but is not completely forgotten. An input space which satisfies this property will be referred to as an *input space with infinite memory.*

If the mapping F from U into Y (or from U^e into Y^e) is such that $\|u'_t-u''_t\|_t=0$ implies $|||F(u'_t)-F(u''_t)|||_t=0$ then a mapping $\tilde{F}_t$ (called a truncated system mapping) is defined from U_t into Y_t. Under this condition we say (Y,F,U) is a *causal system.* Clearly it is necessary in order for a causal system to be defined that F be causal in the usual sense. Sometimes it is also sufficient, but not always. If U has "bounded memory"; i.e., $\|u_t\|_t=\|u_t\|_{t-d,t}$ for some fixed $0<d<\infty$, for all u and all t, this condition can be violated; (Y,F,U) may *not* be a causal system by our definition, even though F "looks like" a causal operator. This does not happen for any fitted family where $\|u_t\|_t=0$ implies $\|u_t\|_s=0$ for all $s\le t$ - the usual case. When a causal system is defined we put $F_t\overset{\Delta}{=}\mathrm{R}_{-t}\tilde{F}_t\mathrm{R}_t$, where R_t denotes right translation by

t. Thus $F_t : U_0 \to Y_0$ determines the behavior of the system at time t. We define a certain Banach space of maps F_t:

Definition 3: Let $F_t : U_0 \to Y_0$ be continuous and such that for some fixed nonnegative integer N,

$$\square F_t \square_0 \overset{\Delta}{=} \sup_{u_0 \in U_0} \frac{|||F_t(u_0)|||_0}{1 + \|u_0\|_0^N} \tag{2}$$

is finite.

The set of all F_t satisfying (2) is a Banach space under the usual linear operations, with norm as shown. This space is denoted $C_N^\infty(-\infty,0]$. We define a Banach space for the maps $F : U \to Y$ by $\square F \square \overset{\Delta}{=} \sup_t \square F_t \square_0$. This space is denoted $C_N^\infty(-\infty,\infty)$. When $N=0$ C_N^∞ is trivially equivalent to the well known Banach space C^∞ of continuous operators with bounded range (used in [6], [7].)

The feedback system considered consists of a single loop with an input-output system in the forward and in the feedback paths. The feedback equations are

$$\begin{cases} u_1 = e_1 + He_2 \\ u_2 = e_2 - Ge_1 \end{cases} . \tag{3}$$

The feedback system has two inputs u_1 and u_2 and two outputs $y_1(=Ge_1)$ and $y_2(=He_2)$. It will be assumed that $A=B$ for both systems so that the two input spaces and the two output spaces are the same spaces. The systems will be referred to as (Y,G,U) and (Y,H,U).

It will be necessary that U have the property that it permits the splicing of inputs. Roughly speaking, if $u,v \in U$, the splice of u and v at time s, denoted $u_s \leftrightarrow v_{s,\infty}$, is the element of U that agrees with u for time $t \le s$ and with v for $t > s$ (see [5] or [6].)

BOUNDEDNESS AND EXISTENCE THEOREMS

Three fundamental feedback theorems that can be found in Chapter III of [4] will be presented in the context of fitted families, which will allow the use of weighted (and in particular tapered) input and output spaces. Also, the time axis will be the entire real line, as opposed to [4] and [9] for example, where it is only $[0,\infty)$. The first two results are stated without proof, since their proofs would follow step by step the proofs given in [4] of the original theorems. The third result requires the assumption that the input space is a zero-shift space and a proof is provided.

Proposition 1: (Small gain theorem.) Consider the feedback system defined by (3). If

$$\|Ge_1\|_t \le \gamma_1 \|e_1\|_t + \beta_1$$

$$\|He_2\|_t \le \gamma_2 \|e_2\|_t + \beta_2$$

for all $e_1, e_2 \in U^e$ and all $t \in I\!R$ where $\gamma_1, \gamma_2, \beta_1, \beta_2$ are constants and such that $\gamma_1 \ge 0, \gamma_2 \ge 0$ and $\gamma_1\gamma_2 < 1$; then,

$$\|e_1\|_t \le (1-\gamma_1\gamma_2)^{-1}\{\|u_1\|_t + \gamma_2\|u_2\|_t + \beta_1 + \gamma_2\beta_1\} \tag{4}$$

$$\|e_2\|_t \le (1-\gamma_1\gamma_2)^{-1}\{\|u_2\|_t + \gamma_1\|u_1\|_t + \beta_2 + \gamma_1\beta_2\} \quad . \tag{5}$$

Equations (4) and (5) imply that if $u_1, u_2 \in U$ then $e_1, e_2 \in U$ (the bounding space.) Causality and existence of a solution to the feedback equations (3) are not implied. In Proposition 1 it is not necessary that U_t be complete.

Proposition 2: (Small gain theorem - incremental form.) Consider the feedback system defined by (3). Assume that for all $u_1, v_1, u_2, v_2 \in U^e$

$$\|Gu_1 - Gv_1\|_t \le \gamma_1 \|u_1 - v_1\|_t$$

$$\|Hu_2 - Hv_2\|_t \le \gamma_2 \|u_2 - v_2\|_t$$

for all t, where γ_1 and γ_2 are positive constants such that $\gamma_1\gamma_2 < 1$. Assume that U_t is a Banach space for all

t. Then for all $u_1,u_2 \in U$, respectively $\in U_t$, there exists a unique solution $e_1,e_2 \in U$, respectively $\in U_t$, to (3) and the map $(u_1,u_2) \to (e_1,e_2)$, respectively $(u_{1t},u_{2t}) \to (e_{1t},e_{2t})$, is uniformly continuous.

The following Lemma will be needed in Proposition 3:

Lemma 1: Consider a fitted family of left extended normed linear spaces $\{A_{s,t}\}$. Let splicing be permitted. Given $-\infty < s \le t < \infty$ and $v_s \in U_s$, define a subset of A_t by

$$V(v_s)_t \overset{\Delta}{=} \{v_s \mapsto u_{s,t} \mid u_{s,t} \in A_{s,t}\} .$$

If $A_{s,t}$ is a Banach space then $V(v_s)_t$ is complete.

For the last Proposition in this section, the feedback system is described as follows: Define $U^C = U \times U$ with elements $u=(u_1,u_2)$ and $\|u\| \overset{\Delta}{=} \|u_1\| + \|u_2\|$. Define $G^C e = (He_2, -Ge_1)$. Then the feedback system given by (3) may be expressed as

$$u = e + G^C e \;. \tag{6}$$

Proposition 3: (An existence and uniqueness theorem.) Consider a feedback system defined by (6) with G^C causal and defined on the expanded space $(U^C)^e$. Let $(U^C)^e$ be a zero shift space of time functions that permits splicing. Assume that $U^C{}_{s,t}$ is a Banach space for all $-\infty < s \le t < \infty$. Assume $G^C(0_t)=0_t$ for all $t \in \mathbb{R}$. If for all compact intervals $I \subset \mathbb{R}$ there exists $\Delta(I) > 0$ and $\gamma = \gamma(I) < 1$ such that

$$\|G^C e - G^C f\|_{s,t} \le \gamma\{\|e-f\|_s + \|e-f\|_{s,t}\} \tag{7}$$

where $s,t \in I$, $t-s \le \Delta(I)$ then for any $u \in (U^C)^e$ there exists a unique solution $e \in (U^C)^e$ to the feedback system.

Proof: Given $u \in (U^C)^e$, there exists T such that $u_T = 0_T$. Take I to be a compact interval containing T and such that T is not the right endpoint of I. For the corresponding $\Delta(I)$, let δ, $0 < \delta < \Delta$, be such that $T+\delta \in I$. Then, the equation

$$e_{T+\delta} = u_{T+\delta} - G^C(e_{T+\delta})$$

is in the form $e_{T+\delta} = F(e_{T+\delta})$. It will be shown that F is a contraction on $V(0_T)_{T+\delta}$, (defined in Lemma 1), so that a unique solution (in $V(0_T)_{T+\delta}$) to (6) exists up to time $T+\delta$. Let $e', e'' \in V(0_T)_{T+\delta}$:

$$\|F(e'_{T+\delta}) - F(e''_{T+\delta})\|_{T+\delta} = \|u_{T+\delta} - G^C(e'_{T+\delta}) - u_{T+\delta} + G^C(e''_{T+\delta})\|_{T+\delta}$$

$$\le \|G^C(e'_{T+\delta}) - G^C(e''_{T+\delta})\|_T + \|G^C(e'_{T+\delta}) - G^C(e''_{T+\delta})\|_{T,T+\delta} \;. \tag{8}$$

The first term on the right side of (8) is zero since $e', e'' \in V(0_T)_{T+\delta}$, by the causality of G^C. By (7), the second term is less than or equal to

$$\gamma(I)\{\|e'-e''\|_T + \|e'-e''\|_{T,T+\delta}\} \le \gamma\|e'-e''\|_{T+\delta} \;.$$

Thus, F is a contraction mapping on $V(0_T)_{T+\delta}$ so there is a unique solution to (6) up to time $T+\delta$ in this space. The same calculation may be used to extend the solution to cover the entire real line. Denote the right endpoint of I by b_1. Recall that $T < b_1$. If $T+\delta < b_1$ the solution (within $V(0_T)_{T+\delta}$) can be extended beyond $T+\delta$ as follows: Choose δ_1 such that $T+\delta+\delta_1 \le b_1$ and $0 < \delta_1 < \Delta$. Consider the complete metric space $V(0_T \mapsto u_{T,T+\delta})_{T+\delta+\delta_1}$ where $u_{T,T+\delta}$ is the previously determined increment. The contraction mapping theorem may be used on $V(0_T \mapsto u_{T,T+\delta})_{T+\delta+\delta_1}$ as before. Let $e', e'' \in V(0_T \mapsto u_{T,T+\delta})_{T+\delta+\delta_1}$:

$$\|F(e'_{t+\delta+\delta_1}) - F(e''_{T+\delta+\delta_1})\|_{T+\delta+\delta_1} \le \|G^C(e'_{T+\delta+\delta_1}) - G^C(e''_{T+\delta+\delta_1})\|_T$$

$$+ \|G^C(e'_{T+\delta+\delta_1}) - G^C(e''_{T+\delta+\delta_1})\|_{T,T+\delta}$$

$$+ \|G^C(e'_{T+\delta+\delta_1}) - G^C(e''_{T+\delta+\delta_1})\|_{T+\delta,T+\delta+\delta_1} \le \gamma(I)\|e'-e''\|_{t+\delta,T+\delta+\delta_1} \;.$$

In this way the solution can be extended to b_1 in a finite number of steps. Since b_1 can be chosen arbitrarily large, a unique solution within $V(0_T)_t$ can be determined for any input, for t arbitrarily large. To see that a unique solution within $(U^C)^e$ is generated, assume that two solutions exist. The two solutions will agree for all $t \le T'$ for some sufficiently negative T' since $(U^C)^e$ is a zero shift space. Although the point T' may be further back than T, the contraction mapping theorem may be invoked once again exactly as above, giving that the two solutions cannot be different. ////

Many results of the same nature as these three theorems or which result from these theorems may also be generalized to hold for tapered input spaces or, as the case may be, defined over the entire real line. For example, all of the results in [1] can be so generalized.

CAUSALITY THEOREMS

Two causality results will be presented. We remark that these results (without further hypotheses) do *not* imply the existence of a solution to the feedback equation (6). However, if a solution exists, i.e. if given u there is defined a corresponding e, then if causality conditions are satisfied the map $\bar{F}_t$ carrying u_t into e_t gives a causal system. Thus there is an implication that causality implies uniqueness; the definition of causality used requires the existence of maps F_t, which in turn implies uniqueness. Consideration to hypotheses which specifically guarantee causality (uniqueness) has not been given by others, as far as we know. The first result shows that if a non-zero finite time delay is present in the feedback loop and the input space is a zero-shift space with infinite memory the feedback system is causal. If the restriction that $G(0)=H(0)=0$ is added a unique solution *exists* for each input. This kind of result is not completely unknown, see [9], Chapter 4.

Proposition 4: Assume that the feedback system described by (6) contains a finite time delay in the feedback loop; i.e., $G^C = \mathrm{R}_D \hat{G}^C$, where the right shift produces a time delay of length D, where $0<D<\infty$ and $\hat{G}^C : (U^C)^e \to (U^C)^e$ is a causal input-output system. Assume further that the input space $(U^C)^e$ is a zero shift input space and is an input space with infinite memory. Then if there is a solution to the feedback equation in $(U^C)^e$, the feedback system is causal.
Proof: To prove causality, let $e', e'' \in (U^C)^e$ be such that $\|e'-e''\|_t \neq 0$ for some arbitrary $t \in \mathbb{R}$. Because $(U^C)^e$ is a zero shift space, there exist $s<t$ and $0<\sigma<D$ with $s+\sigma \leq t$ such that $\|e'-e''\|_s = 0$ and $\|e'-e''\|_{s,s+\sigma} \neq 0$. Then defining $u'=(I+G^C)e'$ and $u''=(I+G^C)e''$,

$$\|u'-u''\|_{s+\sigma} \geq \|u'-u''\|_{s,s+\sigma} = \|(I+\mathrm{R}_D\hat{G}^C)e'-(I+\mathrm{R}_D\hat{G}^C)e''\|_{s,s+\sigma}$$

$$\geq \|e'-e''\|_{s,s+\sigma} - \|\mathrm{R}_D\hat{G}^C e' - \mathrm{R}_D\hat{G}^C e''\|_{s,s+\sigma}$$

$$= \|e'-e''\|_{s,s+\sigma} - \|\hat{G}^C e' - \hat{G}^C e''\|_{s-D,s+\sigma-D} \quad . \tag{9}$$

The second term in (9) is zero since $(U^C)^e$ is a space with infinite memory (giving $\|e'-e''\|_{s+\sigma-D}=0$) and $\hat{G}^C$ is causal. Considering that the first term in (9) is not zero so that $\|u'-u''\|_{s+\sigma} \neq 0$, the hypotheses on $(U^C)^e$ gives that $\|u'-u''\|_t \neq 0$. Therefore, if a solution to the feedback system exists, it is causal. ////

Corollary 1: Under the hypotheses of Proposition 4, if in addition $(G^C(0))_t = 0_t$, then for all $u \in (U^C)^e$ there exists a unique (causal) solution to the feedback system.
Proof: Given $u \in (U^C)^e$ a unique e and hence y can be determined in the following manner. The feedback equation is rewritten to emphasize the causality of the solution:

$$u_t = (I+G^C)e_t \quad . \tag{6'}$$

There exists $T \in \mathbb{R}$ such that $u_T=0$. From (6') it can be seen that $e_t=0$ for $t \leq T$. For $t>T$, e_t is determined by tracing around the feedback loop using the time delay. To be more explicit, since a time delay is present in the loop, for any $t \in \mathbb{R}$ the input up to time $t+D$ and the error up to time t determine the error up to time $t+D$. Hence, a solution can be constructed, by Proposition 4 it is unique. ////

The Proposition to follow considers only the causality of the feedback system. Part of its hypothesis is that the mapping $t \to \|u\|_t$ is continuous. A sufficient condition for this is that $(U^C)^e$ is expanded shift continuous; i.e., for all $u \in (U^C)^e$, $-\infty \leq s \leq t < \infty$, $-\infty < t$, $\lim\limits_{h \to 0} \|\mathrm{L}_h u - u\|_{s,t} = 0$. Weighted L_p spaces, $p<\infty$ are shift continuous.

Proposition 5: Consider the feedback system defined by (6). Assume that $(U^C)^e$ is a zero shift input space with infinite memory and such that $t \to \|u\|_t$ is continuous. Assume that for $u', u'' \in (U^C)^e$,

$$\varlimsup_{\tau \to \sigma} \frac{\|G^C(u')-G^C(u'')\|_{\sigma,\tau}}{\|u'-u''\|_{\sigma} + \|u'-u''\|_{\sigma,\tau}} < 1 \tag{10}$$

for all $\sigma \in \mathbb{R}$, where $\tau \geq \sigma$. Then if there exists a solution to the feedback equation it is a causal solution.
Proof: Again, causality is shown by proving the contrapositive. Let the pair (u', e') and the pair (u'', e'') belong to the extended space $(U^C)^e$ and satisfy (6), and assume that $\| e' - e'' \|_t \neq 0$. Because of the conditions on $(U^C)^e$ it is possible to find $s < t$ such that $\| e' - e'' \|_s = 0$ and $\| e' - e'' \|_{s+\delta} \neq 0$ for all $0 < \delta < c$, where $0 < c < (t-s)$. Then

$$\| u' - u'' \|_{s+\delta} = \| e' + G^C e' - e'' - G^C e'' \|_{s+\delta}$$

$$\geq \| e' - e'' \|_{s+\delta} - \{ \| G^C e' - G^C e'' \|_s + \| G^C e' - G^C e'' \|_{s,s+\delta} \}$$

$$= \| e' - e'' \|_{s+\delta} - \| G^C e' - G^C e'' \|_{s,s+\delta} . \tag{11}$$

For some δ small enough, there exists $0 < (1-p) < 1$ such that by using (10) it is seen that (11) is bounded below by

$$\| e' - e'' \|_{s+\delta} - (1-p) \{ \| e' - e'' \|_s + \| e' - e'' \|_{s,s+\delta} \} = \| e' - e'' \|_{s+\delta} - (1-p) \| e' - e'' \|_{s,s+\delta}$$

$$\geq \| e' - e'' \|_{s,s+\delta} - (1-p) \| e' - e'' \|_{s,s+\delta} = p \, \| e' - e'' \|_{s,s+\delta} \neq 0 .$$

Hence, $\| e' - e'' \|_t \neq 0$ implies $\| u' - u'' \|_t \neq 0$; therefore, the feedback system is causal. ////

STABILITY OF LINEAR SYSTEMS WITH EXPONENTIALLY TAPERED INPUT AND OUTPUT SPACES

In this section the effect on the stability of linear systems from the use of exponentially tapered input and output spaces will be considered. Two types of stability will be considered: BIBO stability and bounding space stability. Results such as those to be obtained may be applied to feedback systems and may be used to recover the circle and Popov criteria in some form.

In what follows $U = Y$ is a bounding space as described below; its elements are measurable functions (actually equivalence classes of functions) from $\mathbb{R}$ into E where $E = \mathbb{R}^n$ equipped with norm $\| x \|$, $x \in E$. The space U is equal to A, where A is the bounding space for a fitted family of normed linear spaces $\{A_{s,t}\}$ with norms given by (1) with $1 \leq p < \infty$. For Proposition 6 and the preparatory inequality (14), the norms $\| u \|_{s,t}$ are given by the special form:

$$\| u \|_{s,t} = \left[\int_s^t \| u(\tau) \|^p e^{a(t-\tau)} d\tau \right]^{1/p} , \tag{12}$$

with $-\infty < a \leq 0$. When $a < 0$, the conditions of Definition 1 are satisfied so a tapered fitted family is described. Throughout, the mapping $F : U \to Y$ (or $U^e \to Y^e$) is a linear time-invariant mapping given by a convolution:

$$[F(u)](t) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} g(t-\tau) u(\tau) d\tau , \tag{13}$$

where the kernel g satisfies $g(s) = 0$ for $s < 0$. At one point the system (Y, F, U) will be required to be of *finite order*, i.e. the convolution (13) represents a solution of a linear, constant-coefficient differentiable equation of finite order.

The following preliminary fact and Lemma will be needed. Let $\| \cdot \|_{(L_p)s,t}$ represent the flat L_p norm over $[s, t]$; i.e., $w(t) = 1$ in (1). Let $f \in A^e$ be such that $f(t) = 0$ for $t < -\Delta$, $t > 0$ where $\Delta \geq 0$ is fixed. Then

$$\| f \|_{(L_p) - \Delta, 0} e^{a(\sigma + \Delta)/p} \leq \| f \|_{-\Delta, \sigma} \leq \| f \|_{(L_p) - \Delta, 0} e^{a\sigma/p} , \tag{14}$$

for all $\sigma \geq 0$.

Lemma 2: Consider a fitted family of seminorms as defined by (12). Given $\lambda \in \mathbb{R}$ and $w > 0$ there exists $0 < k < \infty$ such that

$$\| e^{\lambda t} \sin \omega t \|_{s,t} \geq k \, \| e^{\lambda \tau} \|_{s,t} \tag{15}$$

for all $s, t \in \mathbb{R}$, $s \leq t$ where $k = k(\lambda, \omega, (t-s))$ with $k(\lambda, \omega, \Delta_1) > k(\lambda, \omega, \Delta_2)$ for $\Delta_1 > \Delta_2$.

Definition 4: A causal system (Y, F, U) is *BIBO stable* if for all $0 < m < \infty$ there exists $0 < M < \infty$ such that

$\| u \|_t < m$ implies $||| F(u) |||_t < M$, for all $u \in U$ and $t \in \mathbb{R}$.

In the following δ_{-1} will denote the unit step function.

Proposition 6 : Consider a linear time-invariant system (Y, F, U) as defined by (12) and (13) and the accompanying conditions with $1 \le p < \infty$. If $\tilde{g} \overset{\Delta}{=} g \cdot e^{-a(\cdot)/p}$ is absolutely integrable the system is BIBO stable. If the system is of finite-order the converse is also true.
Proof: (1.) $\tilde{g} \in L_1 \Rightarrow$ BIBO stability. Assume $1 < p < \infty$. Let $u_t \in U_t$ and

$$\int_{-\infty}^{\infty} \square g(\tau) e^{-a\tau/p} \square \delta_{-1}(\tau) d\tau < \infty \ ,$$

where $\square \cdot \square$ represents a matrix norm compatible with $\| \cdot \|$ in $\mathbb{R}^n$. Then,

$$\| y \|_t = \left[\int_{-\infty}^{\infty} \| y(s) \|^p e^{a(t-s)} \delta_{-1}(t-s) ds \right]^{1/p}$$

$$= \left[\int_{-\infty}^{\infty} \| \int_{-\infty}^{\infty} g(s-\tau) \delta_{-1}(s-\tau) u(\tau) d\tau \|^p e^{a(t-s)} \delta_{-1}(t-s) ds \right]^{1/p}$$

$$\le \left[\int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} \square g(s-\tau) \square \cdot \| u(\tau) \| e^{a(t-s)/p} \delta_{-1}(s-\tau) d\tau \right]^p \delta_{-1}(t-s) ds \right]^{1/p}$$

$$= \Bigl(\int_{-\infty}^{\infty} \Bigl(\int_{-\infty}^{\infty} \square g(s-\tau) \square^{1/q} e^{-a(s-\tau)/pq} \cdot \square g(s-\tau) \square^{1/p} e^{-a(s-\tau)/p^2} \delta_{-1}(s-\tau)$$

$$\times \ \| u(\tau) \| e^{a(t-\tau)/p} \delta_{-1}(t-\tau) d\tau \Bigr)^p ds \Bigr)^{1/p} ,$$

where $1/p + 1/q = 1$. By Holder's inequality the above expression is dominated by

$$\Bigl(\int_{-\infty}^{\infty} \Bigl(\int_{-\infty}^{\infty} \square g(s-\sigma) \square e^{-a(s-\sigma)/p} \delta_{-1}(s-\sigma) d\sigma \Bigr)^{p/q}$$

$$\times \ \Bigl(\int_{-\infty}^{\infty} \square g(s-\tau) \square e^{-a(s-\tau)/p} \delta_{-1}(s-\tau) \| u(\tau) \|^p e^{a(t-\tau)} \delta_{-1}(t-\tau) d\tau \Bigr) ds \Bigr)^{1/p} ,$$

$$= \square \tilde{g} \square_{(L_1)}^{1/q} \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \square g(s-\tau) \square e^{-a(s-\tau)/p} \delta_{-1}(s-\tau) ds \ \| u(\tau) \|^p e^{a(t-\tau)} d\tau \right]^{1/p}$$

$$= \square \tilde{g} \square_{(L_1)}^{1/q} \cdot \square \tilde{g} \square_{(L_1)}^{1/p} \| u \|_t = \square \tilde{g} \square_{(L_1)} \cdot \| u \|_t \ . \tag{16}$$

Hence, the system is BIBO stable. Since equation (16) has a finite value, Tonelli's theorem may be invoked to allow the interchange in the order of integration done in the derivation. An obvious modification of this proof yields the result for $p=1$.
(2.) For (Y, F, U) of finite order, BIBO stability $\Rightarrow \tilde{g} \in L_1$. In this case the kernel g consists of a sum of exponentials at the natural frequencies of the system each multiplied by a polynomial in t. Let λ^* be the natural frequency with most positive real part. Given $\delta > 0$, there exists $u \in U$ satisfying $u(t) = 0$ for $t < -\delta$, $t > 0$ such that

$$y(t) = e^{\mathrm{Re.}\lambda^* t} \sin(\mathrm{Im.}\lambda^* t + \theta) \cdot \eta \tag{17}$$

for $t \ge 0$, where θ is some phase angle and η is some constant n-vector. Computing a lower bound for the gain gives for $\sigma > 0$:

$$\frac{||| y |||_\sigma}{\| u \|_\sigma} \ge \frac{||| y |||_{0,\sigma}}{\| u \|_{-\delta,\sigma}}$$

$$\ge \frac{k \| e^{\mathrm{Re.}\lambda^* t} \|_{0,\sigma} \cdot \| \eta \|}{\| u \|_{-\delta,\sigma}} = \frac{k' \left[\int_0^{\sigma} e^{p \cdot \mathrm{Re.}\lambda^* t} e^{a(\sigma-t)} dt \right]^{1/p}}{\| u \|_{-\delta,\sigma}} ,$$

by Lemma 2 and (17), where k is also defined by that Lemma. By (14) the last expression dominates

$$\frac{k'\cdot e^{a\sigma/p}\left[\int_0^{\sigma} e^{p\cdot\mathrm{Re}.\lambda^* t}\, e^{-at}\,dt\right]^{1/p}}{e^{a(\sigma+\delta)/p}\,\|u\|_{(L_p)-\delta,0}}$$

$$=\frac{k'}{e^{a\delta/p}\,\|u\|_{(L_p)-\delta,0}}\left[\int_0^{\sigma} e^{(p\cdot\mathrm{Re}.\lambda^*-a)t}\,dt\right]^{1/p} . \tag{18}$$

From (18), a necessary condition for the BIBO stability of the system is $p\cdot\mathrm{Re}.\lambda^*<a$. However, $p\cdot\mathrm{Re}.\lambda^*<a$ is obviously sufficient for the absolute integrability of $\tilde{g}$. *////*

A possible interpretation of Proposition 6 can be that the system is stable if and only if the memory of the system is "worse" that the memory of the norm on the input space. Proposition 6 is an extension of the usual result for the flat L_p norm. A weaker, less familiar type of stability than BIBO stability is:

Definition 5: A causal mapping $F:U^e\to B^e$ is *bounding space stable* if F maps $u\in U$ to $y\in Y$, where U and Y are the respective bounding spaces.

Proposition 7: Consider an input-output system $F:U^e\to Y^e$ described by (13) with a non-flat fitted family of seminorms defined by (1). If the kernel g is absolutely integrable then the system is bounding space stable.
Proof: Consider

$$\|y\|_t=\left[\int_{-\infty}^{\infty}\|y(s)\|^p w(t-s)ds\right]^{1/p}$$

$$=\left[\int_{-\infty}^{\infty}\left\|\int_{-\infty}^{\infty} g(\tau)u(s-\tau)d\tau\right\|^p w(t-s)ds\right]^{1/p}$$

$$\le\left[\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\Box g(\tau)\Box\cdot\|u(s-\tau)\|d\tau\right]^p w(t-s)ds\right]^{1/p}$$

$$=\left[\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\Box g(\tau)\Box^{1/q}\cdot\Box g(\tau)\Box^{1/p}\cdot\|u(s-\tau)\|d\tau\right]^p w(t-s)ds\right]^{1/p}=\mathrm{I} .$$

where $1/p+1/q=1$. Then,

$$\mathrm{I}\le\left[\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\Box g(\sigma)\Box d\sigma\right]^{p/q}\cdot\left[\int_{-\infty}^{\infty}\Box g(\tau)\Box\cdot\|u(s-\tau)\|^p d\tau\right]w(t-s)ds\right]^{1/p}$$

$$=\Box g\Box_{(L_1)}^{1/q}\cdot\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\Box g(\tau)\Box\cdot\|u(s-\tau)\|^p w(t-s)d\tau ds\right]^{1/p}$$

$$=\Box g\Box_{(L_1)}^{1/q}\cdot\left[\int_{-\infty}^{\infty}\Box g(\tau)\Box\cdot\int_{-\infty}^{\infty}\|u(s-\tau)\|^p w(t-s)dsd\tau\right]^{1/p} ,$$

using the Fubini theorem. The above equation becomes

$$=\Box g\Box_{(L_1)}^{1/q}\cdot\left[\int_{-\infty}^{\infty}\Box g(\tau)\Box\cdot\int_{-\infty}^{\infty}\|\mathrm{R}_\tau u(s)\|^p w(t-s)dsd\tau\right]^{1/p} .$$

Note that $(\mathrm{R}_s u)_t\in U_t^e$. Hence, it follows that if $u\in U$

$$\|y\|\le\Box g\Box_{(L_1)}^{(1/q+1/p)}\cdot\|u\|=\Box g\Box_{(L_1)}\cdot\|u\| . \tag{19}$$

Therefore, the system is bounding space stable. *////*

A comment concerning Proposition 7: (19) can be sharpened to

$$\|y\|_t\le\Box g\Box_{(L_1)}\cdot\sup_{s\le t}\|u\|_s . \tag{20}$$

It is (even more) clear from (20) that g absolutely integrable does not imply BIBO stability. Also, (20) does show that for the case where flat seminorms are used g absolutely integrable implies BIBO stability since $\|u\|_t=\sup_{s\le t}\|u\|_s$ if and only if $w(t)=1$ for all $t\ge 0$.

REFERENCES

[1] V. Anantharam and C.A. Desoer, "On the Stabilization of Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. AC-29, No. 6, pp. 569-572, June 1984.

[2] S. Boyd and L.O. Chua, "Fading Memory and the Problem of Approximating Nonlinear Operators with Volterra Series," *IEEE Transactions on Circuits and Systems*, Vol. CAS-32, No. 11, November 1985.

[3] B.D. Coleman and V.J. Misel, "Norms and Semi-Groups in the Theory of Fading Memory," *Archives Rational Mechanics Analysis*, 23: 87-123 1966.

[4] C.A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, 1975.

[5] W.L. Root and D. Serakos, "On Causal Input-Output Systems," *Proceedings 23rd Allerton Conference on Communications, Control and Computing*, 1985.

[6] W.L. Root, "A Note on State Trajectories of Causal Input-Output Systems," *Proceedings 21st Allerton Conference on Communications, Control and Computing*, 1983.

[7] W.L. Root, "Considerations Regarding Input and Output Spaces for Time-Varying Dynamical Systems," *Applied Mathematics and Optimization*, 4: 365-384, 1978.

[8] W.L. Root, "The Identifiability of Abstract Systems," *Tech. Report, in preparation, The University of Michigan.*

[9] J.C. Willems, *The Analysis of Feedback Systems*, The MIT Press, 1971.

PARALLEL GRADIENT PROJECTION ALGORITHMS TO SOLVE THE DISCRETE LQR OPTIMAL CONTROL PROBLEM WITH HARD CONTROL BOUNDS

GERARD G. L. MEYER
Electrical and Computer Engineering Department
The Johns Hopkins University
Baltimore, Maryland 21218

LOUIS J. PODRAZIK
Bendix Environmental Systems Division
Allied-Signal Inc.
Baltimore, Maryland 21284-9840

ABSTRACT

In this paper we present two parallel gradient based iterative algorithms to solve the linear quadratic regulator (LQR) optimal control problem with hard control bounds. In the first part of the paper, we introduce the algorithms in the context of the general class of problems to which they are applicable. The first algorithm is a parametrized gradient projection method and can be used to solve any convex programming problem. The second algorithm is a combination of the first algorithm with a constrained version of the Fletcher-Reeves conjugate gradient method and can be used to solve linear inequality constrained problems. We then use the two algorithms to solve the LQR optimal control problem with hard control bounds. In the second part of the paper, we show that at each iteration parallel evaluation of the step length and projected gradient of the quadratic cost function can be efficiently performed as a function of the number of processors. We then embed our parallel step length and gradient projection procedures to produce two parallel algorithms which are suitable for real-time online implementation on a SIMD machine.

I. INTRODUCTION

Practical iterative methods to solve optimal control problems must exhibit fast convergence coupled with low computational overhead per iteration so that they may be implemented in a real-time online environment. Furthermore, they must be interior descent methods since infeasible approximations to the solution are unacceptable for most applications. In this paper we present two gradient based iterative interior methods for the parallel solution of the discrete LQR optimal control problem with hard control bounds. The algorithms are synthesized so that each iteration may be efficiently executed on a parallel computer. Unlike previous parallel approaches to the solution of optimal control problems which simply fold the computations to fit the number of processors [MEY73], [SCH81], [TRA80], our approach has been devised by explicitly considering the given computational environment. Consequently, one of the features of the parallel algorithms presented in this paper is that their structure, and hence their parallelism, is determined by the number of available processors, resulting in algorithms which are matched to the given parallel environment.

We introduce the algorithms in the context of the general class of problems to which they are applicable. The first algorithm is a parametrized gradient projection method and can be used to solve any convex programming problem. We parametrize the algorithm for two purposes: first, the algorithm contains a so called ε-procedure for determining the active constraints in order to prevent the possibility of jamming and to ensure convergence; unlike approaches which generate a sequence of ε's, our approach uses a constant ε for all iterations. Secondly, we investigate the performance of the algorithm for various parameter choices with the goal of obtaining improved convergence. The second algorithm is a combination of the first algorithm with a constrained version of the Fletcher-Reeves conjugate gradient method and can be used to solve any problem with linear inequality constraints. Furthermore, we show that a slight modification of our second algorithm results in the algorithm exhibiting finite convergence on problems

---

Supported by the Air Force Office of Scientific Research under Contract AFOSR-85-0097.

with quadratic cost.

We then use our algorithms to solve the LQR optimal control problem with hard control bounds which we rewrite as a bounded variable quadratic programming problem with special structure. There exist many interior methods applicable to this class of problems, including feasible directions, gradient projection, reduced gradient, variable metric and numerous quadratic programming methods. Our motivation for devising a gradient projection method is due to the simple projection computation that occurs with respect to the control problem of interest as well as avoiding large matrix computations involving the exact or approximated Hessian.

We give two parallel implementations of our algorithms to solve the LQR optimal control problem with hard control bounds. In both implementations the step length and gradient is obtained by using the parallel procedures presented in [MEY87b]. The procedures share common terms and exhibit varying degrees of parallelism as a function of the number of processors. We constrain the number of available processors, p, to lie in the range $1 \leq p \leq nN^{1/2}$, where n is the size of the system state vector, N is the number of stages in the control process and we assume $n \geq m$, where m is the size of the control.

By employing a blocked formulation of the approach for first-order linear recurrences given in [MEY87a], we embed in our algorithms the approach for gradient computation given in [MEY87b] which reduces to a direct parallel implementation of the technique given in [POL71, pp.66-69] when $1 \leq p \leq n$ and achieves speedup greater than $1/2(p+n)$ when $n < p \leq nN^{1/2}$. In addition, one of the features of our proposed parallel iterative algorithms is that their structure is completely specified by the number of processors whenever the number of stages $N \geq (p/n)^2$.

The organization of this paper is as follows : Section II introduces the convex problem and associated optimality conditions. Section III presents the parametrized gradient projection algorithm and corresponding implementation for the bounded variable quadratic programming problem. In Section IV we combine the algorithm given in Section III with a linearly constrained version of the Fletcher-Reeves conjugate gradient method. We then rewrite the LQR problem with hard control bounds as a bounded variable quadratic programming problem in Section V. Section VI presents the parallel algorithms to solve the LQR problem with hard control bounds and the corresponding performance analysis. Finally, in Section VII conclusions are presented.

II. THE CONVEX PROBLEM AND PRELIMINARIES

We consider the following problem:

Problem 1: Given $m+1$ maps $f^0(.), f^1(.), ..., f^m(.) : E^n \rightarrow E^1$ and a subset Ω of E^n defined by

$$\Omega = \{x \mid f^i(x) \leq 0 \text{ for } i = 1, 2, ..., m\},$$

find a point x^* in Ω such that for every x in Ω

$$f^0(x^*) \leq f^0(x).$$

Definition 1: Let C be the class of all problems of the form of Problem 1 in which the maps $f^0(.), f^1(.), ..., f^m(.)$ are such that

(i) $f^0(.)$ is continuously differentiable, convex and radially unbounded; that is, $f^0(.)$ is such that given any $x \in \Omega$, to every scalar α corresponds a scalar $\rho > 0$ such that $f^0(x) > \alpha$ whenever $\|x\| > \rho$,

(ii) $f^i(.)$, $i = 1, 2, ..., m$, are continuously differentiable, convex and define a nonempty constraint set Ω and

(iii) the set Ω satisfies the Kuhn-Tucker constraint qualification at every solution x^*.

It is known that the following necessary conditions of optimality are also sufficient for any problem in C.

Optimality Conditions: A point $x^* \in \Omega$ is an optimal solution to a problem in C if and only if there exists a vector $\mu \in E^n$ with components $\mu_i \geq 0$, $i = 1, 2, ..., m$, such that

$$\nabla f^0(x^*)^t + \sum_{i=1}^{m} \mu_i \nabla f^i(x^*)^t = 0$$

$$\mu_i f^i(x^*) = 0, \text{ for } i = 1, 2, ..., m.$$

III. THE PARAMETRIZED GRADIENT PROJECTION ALGORITHM

Given a point $x \in \Omega$ and the parameter $\varepsilon \geq 0$, define the ε-active constraints index set $I(x,\varepsilon)$ as

$$I(x,\varepsilon) = \{i \mid f^i(x) \geq -\varepsilon\},$$

given a subset I of the set $\{1, 2, ..., m\}$, define the subset $\Omega(I)$ of E^n as

$$\Omega(I) = \begin{cases} \{y \mid f^i(y) \leq 0 \text{ for all } i \in I\} & \text{if } I \neq \phi \\ E^n & \text{if } I = \phi, \end{cases}$$

and given a point $x \in \Omega$ and an index set I, let $w(x,I)$ be the projection of $x - \nabla f^0(x)^t$ onto the set $\Omega(I)$, that is, $w(x,I)$ satisfies

$$\|(x - \nabla f^0(x)^t) - w(x,I)\| = \min_y\{\|(x - \nabla f^0(x)^t) - y\| \mid y \in \Omega(I)\}.$$

We only consider problems in C, and thus $\Omega(I)$ is non-empty and convex, and the quantity $w(x,I)$ is well defined. Using the notation just defined, we may rewrite the optimality conditions as:

Lemma 1: If a point $x^* \in \Omega$ is optimal for a problem in C then $w(x^*,I(x^*,\varepsilon)) = x^*$ for all $\varepsilon \geq 0$. Conversely, if x^* and ε satisfy $w(x^*,I(x^*,\varepsilon)) = x^*$, then x^* is optimal.

We now give the parametrized gradient projection algorithm to solve a Problem in the class C.

Algorithm 2: Let $x^1 \in \Omega$ and $\varepsilon \geq 0$ be given.

Step 0: Set $k = 1$.
Compute $\nabla f^0(x^1)$, $I(x^1,\varepsilon) = \{i \mid f^i(x^1) \geq -\varepsilon\}$ and $p^1 = w(x^1,I(x^1,\varepsilon)) - x^1$.

Step 1: If $p^k = 0$ stop; else go to Step 2.

Step 2: Set $d^k = p^k$.

Step 3: Compute $\alpha^k > 0$ to minimize $f^0(x^k + \alpha^k d^k)$ subject to $(x^k + \alpha^k d^k) \in \Omega$.

Step 4: Let $x^{k+1} = x^k + \alpha^k d^k$.

Step 5: Compute $\nabla f^0(x^{k+1})$.

Step 6: Compute $I(x^{k+1},\varepsilon) = \{i \mid f^i(x^{k+1}) \geq -\varepsilon\}$.

Step 7: Compute $p^{k+1} = w(x^{k+1},I(x^{k+1},\varepsilon)) - x^{k+1}$.

Step 8: Set $k = k + 1$ and go to Step 1.

Remark 1: Although not explicitly required in the presentation of Algorithm 2, we introduce the quantity p^k in order to facilitate a later modification which results in our second algorithm.

At each iteration, Algorithm 2 generates a search direction d^k which is computed by projecting the negative gradient onto the set $\Omega(I(x^k,\varepsilon))$. The parameter ε defines the "sufficiently-active" constraint region. If a point x^k is in that region, then $I(x^k,\varepsilon) \neq \phi$ and the corresponding search direction d^k is obtained by projecting $x^k - \nabla f^0(x^k)^t$ onto the set $\{y \mid f^i(y) \leq 0 \text{ for all } i \in I(x^k,\varepsilon)\}$; otherwise, $d^k = -\nabla f^0(x^k)^t$. For ε large enough, $\Omega(I(x^k,\varepsilon)) = \Omega$ for all k. Thus, by choosing ε large, Algorithm 2 reduces to projecting $x^k - \nabla f^0(x^k)^t$ onto the entire set Ω at each iteration. For $\varepsilon = 0$, $\Omega(I(x^k,\varepsilon)) = E^n$ whenever $x^k \in \text{interior}(\Omega)$. Thus, when $\varepsilon = 0$, Algorithm 2 reduces to a projected version of best-step steepest descent in which the direction chosen at an interior point is the negative gradient.

We now show that using any $\varepsilon > 0$ in Algorithm 2 will always produce a solution to a problem in C, even though the direction finding map used to compute d^k is not closed.

Lemma 2: If $x \in \Omega$ is not a solution to a given problem in C and $\varepsilon > 0$ then there exists an $\varepsilon(x) > 0$ and $\delta(x) > 0$ such that

$$f^0(y + \alpha d) \leq f^0(y) - \delta(x) \quad \text{for all } y \in B(x, \varepsilon(x)) \cap \Omega .$$

Theorem 1: If $\{x^k\}$ is a sequence constructed by Algorithm 2 to solve a problem in C then either $\{x^k\}$ is finite and the last point is optimal or $\{x^k\}$ is infinite and contains cluster points, each of which is optimal. Furthermore, when a problem in C is such that $f^0(.)$ is strictly convex, then that problem has a unique optimal solution x^*, and in that case $\{x^k\}$ converges to x^*.

In order to present the parallel implementation of Algorithm 2 used to solve the LQR problem, we first use it to solve a particular case of Problem 1, specifically the bounded variable quadratic programming problem given below.

Problem 2: Find $x \in E^n$ with components x_i, $i = 1, 2, ..., n$, that minimizes the performance index $f^0(x) = 1/2\, x^t H x + b^t x + c$ subject to $x \in \Omega$, where H is an $n \times n$ symmetric positive definite matrix, b is an $n \times 1$ vector and Ω is the unit hypercube defined as $\Omega = \{x \mid \ |x_i| \leq 1, \text{ for } i = 1, 2, ..., n \}$.

Since H is positive definite, the cost function $f^0(.)$ is strictly convex. The nonempty constraint set Ω is the intersection of the $2n$ linear inequalities $f^i(x) = x_i - 1 \leq 0$ and $f^{i+n}(x) = -x_i - 1 \leq 0$ for $i = 1, 2, ..., n$. Therefore, Problem 2 lies in C and by Theorem 1, Algorithm 2 will generate a sequence which converges to the unique solution x^*.

We now give some properties of implementing Algorithm 2 to solve Problem 2.

Lemma 3: Given x^k and $\nabla f^0(x^k)$, let $p^k = w(x^k, I(x^k,\varepsilon)) - x^k$ for the case of Problem 2. Then each component p_i^k of p^k can be computed as

$$p_i^k = \begin{cases} +1 - x_i^k & \text{if} \quad i \in I(x^k,\varepsilon) \text{ and } x_i^k - \nabla f_i^0(x^k)^t \geq 1 \\ -1 - x_i^k & \text{if } i + n \in I(x^k,\varepsilon) \text{ and } x_i^k - \nabla f_i^0(x^k)^t \leq -1 \\ -\nabla f_i^0(x^k)^t & \text{otherwise,} \end{cases} \tag{3.1}$$

where we denote $\nabla f_i^0(x^k)$ to be the i-th component of $\nabla f^0(x^k)$.

Remark 2: If $\varepsilon \geq 2$ then $I(x^k,\varepsilon) = \{1, 2, ..., 2n\}$ for all k. Consequently, $w(x^k, I(x^k,\varepsilon)) = SAT(x^k - \nabla f^0(x^k)^t)$ and Algorithm 2 reduces to projecting onto the entire set Ω at each iteration.

Lemma 4: Given x^k and $\nabla f^0(x^k)$, let α^k minimize $f^0(x^k + \alpha^k d^k)$ subject to $(x^k + \alpha^k d^k) \in \Omega$ for the case of Problem 2. Then α^k can be computed as $\alpha^k = \min\{\alpha_u, \alpha_c\}$, where

$$\alpha_u = -\langle d^k, \nabla f^0(x^k)^t \rangle \ / \ \langle d^k, Hd^k \rangle, \tag{3.2}$$

$$\alpha_c = \min_{\alpha} \{\alpha \mid \alpha = \frac{sgn\,(d_i^k) - x_i^k}{d_i^k} \text{ for all } i \text{ such that } d_i^k \neq 0\}. \tag{3.3}$$

Remark 3: Since the cost function is quadratic and we are using the update $x^{k+1} = x^k + \alpha^k d^k$, the quantity $\nabla f^0(x^{k+1})^t$ can be updated using $\nabla f^0(x^{k+1})^t = \nabla f^0(x^k)^t + \alpha^k H d^k$. Consequently, we need only compute the gradient directly in Step 0 of Algorithm 2, and then use the gradient update at each subsequent iteration.

We now give the following implementation of Algorithm 2 to solve Problem 2.

Algorithm 3: Let x^1 and $\varepsilon > 0$ be given.

Step 0: Set $k = 1$.
Compute $\nabla f^0(x^1)^t = Hx^1 + b$, $I(x^1,\varepsilon) = \{i \mid x_i^1 - 1 \geq -\varepsilon\} \cup \{i + n \mid -x_i^1 - 1 \geq -\varepsilon\}$ and p^1 using Eq. (3.1).

Step 1: If $\|p^k\|^2 \leq \varepsilon$, then stop; else go to Step 2.

Step 2: Set $d^k = p^k$.

Step 3: Compute $\alpha^k = \min\{\alpha_u, \alpha_c\}$, according to Eqs. (3.2) and (3.3).

Step 4: Let $x^{k+1} = x^k + \alpha^k d^k$.

Step 5: Let $\nabla f^0(x^{k+1})^t = \nabla f^0(x^k)^t + \alpha^k H d^k$.

Step 6: Compute $I(x^{k+1},\varepsilon) = \{i \mid x_i^{k+1} - 1 \geq -\varepsilon\} \cup \{i+n \mid -x_i^{k+1} - 1 \geq -\varepsilon\}$.

Step 7: Compute p^{k+1} using Eq. (3.1).

Step 8: Set $k = k + 1$ and go to Step 1.

In Section IV we modify Algorithm 3 to obtain improved convergence properties when a problem in C is linearly constrained.

IV. FLETCHER-REEVES MODIFICATION FOR LINEARLY CONSTRAINED PROBLEMS

We now consider the following linearly constrained class of problems.

Definition 2: Let L be the class of all problems of the form of Problem 1 in which the maps $f^0(.), f^1(.), \ldots, f^m(.)$ are such that
(i) $f^0(.)$ is continuously differentiable, convex and radially unbounded,
(ii) $f^i(x) = \langle g^i, x\rangle - h_i$, for $i = 1, 2, \ldots, m$ define a nonempty constraint set Ω and
(iii) for any $x \in \Omega$, g^i for $i \in I(x, 0)$ are linearly independent.

Given $x^1 \in \Omega$, let $I(x^k,0)$ be the index set defined by $I(x^k,0) = \{i \mid \langle g^i, x\rangle - h_i = 0\}$ and let G_k be the corresponding matrix whose rows are $(g^i)^t$ for all $i \in I(x^k,0)$. The projection matrix P_k associated with the point x^k is $P_k = I - G_k^t(G_kG_k^t)^{-1}G_k$. Let $M(I(x^k,0))$ be the linear manifold defined by

$$M(I(x^k,0)) = \begin{cases} \{x \mid \langle g^i, x\rangle - h_i = 0 \text{ for all } i \in I(x^k,0)\} & \text{if } I(x^k,0) \neq \phi \\ E^n & \text{if } I(x^k,0) = \phi. \end{cases}$$

Then we may compute the projection of $\nabla f^0(x^k)$ onto the set $M(I(x^k,0))$ as $\hat{p}^k = -P_k\nabla f^0(x^k)^t$.

Our approach to solving a problem in L is to approximate $f^0(.)$ as a quadratic and generate a new direction d^{k+1} which is conjugate with respect to the previous direction d^k whenever possible. We compute the initial direction $d^1 = p^1$ and subsequent directions d^k using $d^k = p^k + \beta^k d^{k-1}$, where $\beta^k = \|p^k\|^2/\|p^{k-1}\|^2$. However, rather than projecting $x^k - \nabla f^0(x^k)^t$ onto the active contstraint set $M(I(x^k,0))$, we project onto the approximation set $\Omega(I(x^k,\varepsilon))$, that is, we compute $p^k = w(x^k, I(x^k,\varepsilon)) - x^k$. Consequently, we may produce a d^{k+1} which is conjugate to d^k whenever $P_{k+1} = P_k$ and $p^{k+1} = \hat{p}^{k+1}$; otherwise we restart the algorithm with $d^{k+1} = p^{k+1}$. Furthermore, if $f^0(.)$ is not quadratic, the algorithm should be restarted at least every $n - |I(x^{k+1},0)|$ steps as a spacer step to ensure global convergence.

We now give the parametrized gradient projection algorithm with the Fletcher-Reeves modification to solve a problem in the class L.

Algorithm 4: Let x^1 and $\varepsilon > 0$ be given.

Step 0: Set $k = \hat{k} = 1$ and $d^0 = 0$.
Compute $\nabla f^0(x^1)$, $I(x^1,\varepsilon) = \{i \mid f^i(x^1) \geq -\varepsilon\}$, $I(x^1,0) = \{i \mid f^i(x^1) = 0\}$ and $p^1 = w(x^1, I(x^1,\varepsilon)) - x^1$.

Step 1: If $p^k = 0$ then stop; else go to Step 2.

Step 2: If $d^{k-1} = 0$ then let $d^k = p^k$ and go to Step 5; else go to Step 3.

Step 3: Let $\beta^k = \dfrac{\|p^k\|^2}{\|p^{k-1}\|^2}$.

Step 4: Let $d^k = p^k + \beta^k d^{k-1}$.

Step 5: Compute $\alpha^k > 0$ to minimize $f^0(x^k + \alpha^k d^k)$ subject to $(x^k + \alpha^k d^k) \in \Omega$.

Step 6: Let $x^{k+1} = x^k + \alpha^k d^k$.

Step 7: Compute $\nabla f^0(x^{k+1})$.

Step 8: Compute $I(x^{k+1},\varepsilon) = \{i \mid f^i(x^{k+1}) \geq -\varepsilon\}$.

Step 9: Compute $p^{k+1} = w(x^{k+1}, I(x^{k+1},\varepsilon)) - x^{k+1}$.

Step 10: Compute $I(x^{k+1},0) = \{i \mid f^i(x^{k+1}) = 0\}$.

Step 11: Compute $\hat{p}^{k+1} = -P_{k+1}\nabla f^0(x^{k+1})^t$.

Step 12: If $I(x^{k+1},0) \neq I(x^k,0)$ or $p^{k+1} \neq \hat{p}^{k+1}$ or $\hat{k} > n - |I(x^{k+1},0)|$ then set $d^k = \hat{k} = 0$.

Step 13: Set $k = k + 1$, $\hat{k} = \hat{k} + 1$ and go to Step 1.

When $f^0(.)$ is a quadratic, Algorithm 4 generates a new direction d^{k+1} which is conjugate to d^k whenever the point x^{k+1} is in the same active constraint set $M(I(x^k,0)$ as x^k and $p^{k+1} = \hat{p}^{k+1}$. However, since p^{k+1} is obtained by projecting onto the set $\Omega(I(x^{k+1},\varepsilon))$ rather than $M(I(x^k,0))$, Algorithm 4 is not a pure active set method. Thus, Algorithm 4 may change constraint sets before the minimizer on the current constraint set is found and consequently does not exhibit finite convergence in the case of quadratic cost.

Theorem 3: If $\{x^k\}$ is a sequence constructed by Algorithm 4 to solve a problem in L, then either $\{x^k\}$ is finite and the last point is optimal or $\{x^k\}$ is infinite and contains cluster points, each of which is optimal. Furthermore, when a problem in L is such that $f^0(.)$ is strictly convex, then that problem has a unique optimal solution x^*, and in that case $\{x^k\}$converges to x^*.

We now give a slight modification to Algorithm 4 which allows the resulting algorithm to achieve finite convegence when used to solve problems with quadratic cost.

Algorithm 5: Let x^1 and $\varepsilon > 0$ be given.

Step 1: Use a conjugate gradient method to find the minimizer x_u^* for the unconstrained version of the problem.

Step 2: If $x_u^* \in \Omega$ then stop; else go to Step 3.

Step 3: Compute x_p^1 to be the projection of x_u^* onto the set Ω.

Step 4: Use Algorithm 4 with x_p^1 as the starting point to compute the solution of the problem.

Corollary 1: If $\{x^k\}$ is a sequence constructed by Algorithm 5 to solve a problem in L with $f^0(.)$ a quadratic in x with postitive definite Hessian, then $\{x^k\}$ is finite and the last point is optimal.

V. DISCRETE LQR CONTROL PROBLEM WITH HARD CONTROL BOUNDS

In this section we rewrite the discrete LQR optimal control problem with hard control bounds as a bounded variable quadratic programming problem and then show how the gradient and step length computations may be organized to share common terms.

Problem 3: Given an m-input, discrete, time-varying linear system in which we are given the initial state, $z_0 \in E^n$, and

$$z_i = A_i z_{i-1} + B_i u_i, \quad i = 1, 2, ..., N, \tag{5.1}$$

where for $i = 0, 1, ..., N$, $z_i \in E^n$ is the state of the system at time i and for $i = 1, 2, ..., N$, $u_i \in E^m$ is the control at time i with components $u_{i,j}$, $j = 1, 2, ..., m$, find the mN control vector $u = (u_1^t, u_2^t, \ldots, u_N^t)^t$ that minimizes the performance index

$$J(u) = \frac{1}{2} \sum_{i=1}^{N} \left(z_i^t Q_i z_i + u_i^t R_i u_i \right),$$

and satisfies

$$u \in \Omega,$$

where for $i = 1, 2, ..., N$, Q_i are $n \times n$ symmetric positive semi-definite matrices, R_i are $m \times m$ symmetric positive definite matrices and the convex compact set Ω is defined as

$$\Omega = \{u \mid \ |u_{i,j}| \leq 1, \quad \text{for } i = 1, 2, ..., N, \ j = 1, 2, ..., m\}.$$

Using the notation introduced in [MEY86], Problem 3 may be rewritten as a bounded variable quadratic programming problem in u:

Problem 4: Find the control vector $u \in E^{mN}$ that minimizes the performance index $J(u) = 1/2\, u^t H u + b^t u + c$ subject to $u \in \Omega$, where H is the $mN \times mN$ block symmetric positive definite matrix with block size $m \times m$ given by $H = (R + F_u^t F_z^{-t} Q F_z^{-1} F_u)$, b is the $mN \times 1$ vector given by $b^t = z_0^t F_0^t F_z^{-t} Q F_z^{-1} F_u$, and c is the scalar given by $c = 1/2 z_0^t F_0^t F_z^{-t} Q F_z^{-1} F_0 z_0$, and the matrices R, Q, F_z, F_u and F_0 are defined as: $Q = Diag(Q_1, Q_2, ..., Q_N)$, is the $nN \times nN$ block matrix that consists of N $n \times n$ symmetric positive semi-definite blocks Q_i, $R = Diag(R_1, R_2, ..., R_N)$ is the $mN \times mN$ block matrix that consists of N $m \times m$ symmetric positive definite blocks R_i, F_z is the $nN \times nN$ block lower bidiagonal matrix that consists of N^2 $n \times n$ blocks $\left(F_z\right)_{ij}$ and F_u is the $nN \times mN$ block diagonal matrix that consists of N^2 $n \times m$ blocks $\left(F_u\right)_{ij}$ defined for all i and j in $[1,2,...,N]$ by

$$\left(F_z\right)_{ij} = \begin{cases} I & \text{if } i = j \\ -A_i & \text{if } i = j+1, \\ 0 & \text{otherwise} \end{cases} \qquad \left(F_u\right)_{ij} = \begin{cases} B_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

and F_0 is the $nN \times n$ block matrix that consists of N $n \times n$ blocks $\left(F_0\right)_i$ defined for all i in $[1,2,...,N]$ by

$$\left(F_0\right)_i = \begin{cases} A_1 & \text{if } i = 1 \\ 0 & \text{otherwise}. \end{cases}$$

Since Problem 4 is of the form of Problem 2, Algorithms 3 and 4 can be used to solve Problem 4. We now use the approach for the evaluation of $g(u^1)$, α^k and $g(u^{k+1})$ given in [MEY87b] in which α^k and $g(u^{k+1})$ are computed by sharing common terms, where we use the notation $g(u^k) = (dJ/du)^t_{u=u^k}$.

We first consider the computation of $g(u^1)$, which is performed only once in Step 0 of Algorithms 3 and 4. The matrix F_z is non-singular and thus we may write the gradient as $g(u^1) = Ru^1 + F_u^t F_z^{-t} Q z^1$, where z^1 satisfies Eq. (5.1). Let $\lambda \in E^{nN}$ be the costate vector $\lambda = (\lambda_1^t, \lambda_2^t, \ldots, \lambda_N^t)^t$, $\lambda_i \in E^n$, defined by $\lambda = F_z^{-t} Q z$. Then, given u^1 and z_0, the gradient $g(u^1)$ may be obtained by using the three equations

$$F_z z^1 = F_u u^1 + F_0 z_0, \tag{5.2}$$

$$F_z^t \lambda^k = Q z^k, \tag{5.3}$$

$$g(u^1) = Ru^1 + F_u^t \lambda^1. \tag{5.4}$$

As a consequence of Eqs. (5.2), (5.3) and (5.4) we obtain the gradient evaluation technique proposed by Polak [POL71]:

Algorithm 6: Given u^1 and z_0.

Step 1: Compute z^1 such that $F_z z^1 = F_u u^1 + F_0 z_0$.

Step 2: Compute λ^1 such that $F_z^t \lambda^1 = Q z^1$.

Step 3: Compute $g(u^1) = Ru^1 + F_u^t \lambda^1$.

Due to the lower $n \times n$ block bidiagonal structure of F_z, Steps 1 and 2 of Algorithm 6 require procedures for the solution of N stage forward and reverse $n \times n$ block first-order linear recurrences, respectively. Such procedures are discussed in Section VI. Given λ^1, Step 3 computes $g(u^1)$ by computing each of the N uncoupled components $g_i^1 = (dJ/du_i)^t_{u=u^1}$; thus, Step 3 exhibits linear speedup when executed in parallel.

Algorithms 3 and 4 both use the common term $\pi^k = Hd^k$ in the computation of the step length α^k and the next gradient $g(u^{k+1})$. We now show that π^k can be efficiently computed by writing π^k as

$$\pi^k = Rd^k + F_u^t F_z^t Q F_z^{-1} F_u d^k,$$

and using Algorithm 7 below, where we note that instead of requiring F_z^{-1} and F_z^t, we solve linear systems corresponding to F_z and F_z^t.

Algorithm 7: Given d^k.

Step 1: Compute $\mu^k = Rd^k$.

Step 2: Compute $\delta^k = F_u d^k$.

Step 3: Compute w^k such that $F_z w^k = \delta^k$.

Step 4: Compute $\eta^k = Qw^k$.

Step 5: Compute v^k such that $F_z^t v^k = \eta^k$.

Step 6: Compute $\chi^k = F_u^t v^k$.

Step 7: Compute $\pi^k = \mu^k + \chi^k$.

With the exception of Steps 3 and 5, Algorithm 7 computes π^k by executing a series of matrix-vector products followed by a vector sum. Each of the matrix-vector products consists of N uncoupled block matrix-vector products, thus exhibiting linear speedup when implemented in parallel. However, due to the structure of F_z, Steps 3 and 5 require the solution of N stage forward and reverse $n \times n$ block first-order linear recurrences, respectively. As in the case of Algorithm 6, this again suggests the need for parallel procedures to solve linear recurrences.

VI. PARALLEL ALGORITHMS FOR OPTIMAL CONTROL PROBLEMS

The model of SIMD parallel computation that we use consists of a global parallel memory, p parallel processors, and a control unit, where all processors perform the same operation at each time step. We further simplify the model by making the following assumptions:

A1. Each computational operation requires the same amount of time, referred to as a step.

A2. There are no accessing conflicts in global memory.

A3. All initial data resides in global memory.

A4. There is no time required to access global memory.

We use the parallel procedures FORWARD and REVERSE given in [MEY87b] to solve the forward and reverse N stage $n \times n$ block first-order linear recurrence systems that are required by Algorithms 6 and 7. The procedures are blocked versions of the parallel scalar approach given in [MEY87a] and are formulated as a function of the number p of processors so that the algorithm structure is fixed whenever the number of stages $N \geq (p/n)^2$. We then use the parallel procedures GRADIENT and DIRECTION also given in [MEY87b] to obtain parallel implementations of Algorithms 6 and 7.

We next give the following parallel implementation of computing the quanities $p^k = w(u^k, I(u^k, \epsilon)) - u^k$ and $\hat{p}^k = -P_k g(u^k)$.

```
PROCEDURE PROJECTION1($u^k$, $g(u^k)$)
FORALL $i \in \{1,2,...,N\}, j \in \{1,2,...,m\}$ DO IN PARALLEL $p_{i,j}^k := -g_{i,j}(u^k)$;
FORALL $i \in \{1,2,...,N\}, j \in \{1,2,...,m\}$ DO IN PARALLEL
    IF $u_{i,j}^k - 1 \geq -\epsilon$ AND $u_{i,j}^k - g_{i,j}(u^k) > 1$ THEN $p_{i,j}^k := +1 - u_{i,j}^k$;
    IF $-u_{i,j}^k - 1 \geq -\epsilon$ AND $u_{i,j}^k - g_{i,j}(u^k) < -1$ THEN $p_{i,j}^k := -1 - u_{i,j}^k$;
END FORALL
RETURN $p^k$;
END PROCEDURE
```

```
PROCEDURE PROJECTION2($u^k$, $g(u^k)$)
    FORALL $i \in \{1,2,...,N\}, j \in \{1,2,...,m\}$ DO IN PARALLEL $\hat{p}^k := g_{i,j}(u^k)$;
    $I(u^k,0) := \phi$;
    FORALL $i \in \{1,2,...,N\}, j \in \{1,2,...,m\}$ DO IN PARALLEL
        IF $u^k_{i,j} - 1 = 0$ THEN $\hat{p}^k := 0, I(u^k,0) := I(u^k,0) \cup (i,j)$;
        IF $-u^k_{i,j} - 1 = 0$ THEN $\hat{p}^k := 0, I(u^k,0) := I(u^k,0) \cup (i,j)$;
    END FORALL
    RETURN $\hat{p}^k, I(u^k,0)$;
END PROCEDURE
```

We now embed the parallel procedures GRADIENT, DIRECTION, and PROJECTION1 to obtain a parallel implementation of Algorithm 3 to solve Problem 4 and we then give the corresponding number of steps required for one iteration using p processors.

```
PROCEDURE PGPM($z_0$,$u^1$)
    $k := 1$;
    $g(u^1) :=$ GRADIENT($z_0$,$u^1$);
    $p^1 :=$ PROJECTION1($u^1$, $g(u^1)$);
    WHILE $\|p^k\|^2 > \epsilon_g$ DO
        $d^k := p^k$;
        $\pi^k :=$ DIRECTION($d^k$);
        $\alpha^k := \min\{\alpha_u, \alpha_c\}$;
        FORALL $i \in \{1,2,...,N\}$ DO IN PARALLEL $u_i^{k+1} := u_i^k + \alpha^k d_i^k$;
        FORALL $i \in \{1,2,...,N\}$ DO IN PARALLEL $g_i^{k+1} := g_i^k + \alpha^k \pi_i^k$;
        $p^{k+1} :=$ PROJECTION1($u^{k+1}$, $g(u^{k+1})$);
        $k := k + 1$;
    END WHILE
END PROCEDURE
```

Theorem 4: Given z_0, u^1, N, n, m and p such that $p = 1$ or p/n is an integer, the number of steps required by one iteration of procedure PGPM using p processors, $1 \le p \le nN^{1/2}$, is

$$T(N,n,m,p) = \begin{cases} \left\lceil \frac{Nn}{p} \right\rceil (6n+2m-2) + \left\lceil \frac{Nm}{p} \right\rceil (2n+2m+22) - \frac{4n^2}{p} + 4\log_2 p & \text{if } 1 \le p \le n \\ \left\lceil \frac{Nn}{p} \right\rceil (2n+2m-2) + \left\lceil \frac{Nm}{p} \right\rceil (2n+2m+22) + \left\lceil \frac{N-1}{(p/n)^2-1} \right\rceil 8(p-n) + 4\log_2 p & \text{if } n < p \le nN^{1/2}. \end{cases}$$

We next embed the parallel procedures GRADIENT, DIRECTION, PROJECTION1 and PROJECTION2 to obtain the following parallel implementation of Algorithm 4 to solve Problem 4 and we then give the corresponding number of steps required for one iteration.

```
PROCEDURE PGPFRM($z_0$,$u^1$)
    $k := 1$;
    $g(u^1) :=$ GRADIENT($z_0$,$u^1$);
    $p^1 :=$ PROJECTION1($u^1$, $g(u^1)$);
    WHILE $\|p^k\|^2 > \epsilon_g$ DO
        IF $d^{k-1} = 0$ THEN $d^k := p^k$
        ELSE $\beta^k = \|p^k\|^2 / \|p^{k-1}\|^2$
            FORALL $i \in \{1,2,...,N\}$ DO IN PARALLEL $d_i^k := p_i^k + \beta^k d_i^{k-1}$;
        END IF
        $\pi^k :=$ DIRECTION($d^k$);
        $\alpha^k := \min\{\alpha_u, \alpha_c\}$;
        FORALL $i \in \{1,2,...,N\}$ DO IN PARALLEL $u_i^{k+1} := u_i^k + \alpha^k d_i^k$;
        FORALL $i \in \{1,2,...,N\}$ DO IN PARALLEL $g_i^{k+1} := g_i^k + \alpha^k \pi_i^k$;
        $p^{k+1} :=$ PROJECTION1($u^{k+1}$, $g(u^k)+1$);
        $\hat{p}^{k+1}$ := PROJECTION2($u^{k+1}$, $g(u^{k+1})$);
        IF $I(u^{k+1},0) \neq I(u^k,0)$ OR $p^{k+1} \neq \hat{p}^{k+1}$ THEN $d^k$ := 0;
        $k$ := $k + 1$;
    END WHILE
END PROCEDURE
```

Theorem 5: Given z_0, u^1, N, n, m and p such that $p = 1$ or p/n is an integer, the number of steps required by one iteration of procedure PGPFRM using p processors, $1 \leq p \leq nN^{1/2}$, is

$$T(N,n,m,p) = \begin{cases} \left\lceil \frac{Nn}{p} \right\rceil (6n+2m-2) + \left\lceil \frac{Nm}{p} \right\rceil (2n+2m+34) - \frac{4n^2}{p} + 7\log_2 p & \text{if } 1 \leq p \leq n \\ \left\lceil \frac{Nn}{p} \right\rceil (2n+2m-2) + \left\lceil \frac{Nm}{p} \right\rceil (2n+2m+34) + \left\lceil \frac{N-1}{(p/n)^2-1} \right\rceil 8(p-n) + 7\log_2 p & \text{if } n < p \leq nN^{1/2}. \end{cases}$$

The speedup S_p and efficiency E_p for procedures PGPM and PGPFRM can be obtained directly from Theorems 4 and 5. A straightforward computation shows that S_p and E_p are bounded from below by $S_p \geq 0.6p$ and $E_p \geq 0.6$.

VII. CONCLUSIONS

In this paper two parallel algorithms have been presented to solve the discrete LQR optimal control problem with hard control bounds. The algorithms exhibit fast convergence coupled with a high degree of parallelism at each iteration, making them suitable for real-time online implemenation on an SIMD machine. Moreover, the algorithms possess the desirable property that their structure, and hence parallelism, is determined by the number of available processors. Thus, unlike approaches in which the structure of the procedure changes with problem size, the procedures presented in this paper maintain the same computational and interprocessor communication requirements independently of the number of stages in the control problem. Although not considered in this paper, interprocessor communication requirements should not be a critical performance factor in view of the implementation results for parallel linear recurrence solvers presented in [MEY87a].

Finally, we note that the parallel procedures presented in this paper may be used to solve constrained discrete optimal control problems which involve nonquadratic cost and nonlinear dynamics. In that case one uses suitable approximations in which the system dynamics are linearized and the cost is approximated quadratically.

REFERENCES

[MEY73] Meyer, G.G.L., A Segmented Algorithm for Solving a Class of State Constrained Discrete Optimal Control Problems, *Decision and Control Theory Conference,* San Diego, California, 1973, pp. 73-79.

[MEY87a] Podrazik, L.J. and Meyer, G.G.L., A Parallel First-Order Linear Recurrence Solver, *Journal of Parallel and Distributed Computing*, Vol. 4, No. 2, April, 1987, pp. 117-132.

[MEY87b] Podrazik, L.J. and Meyer, G.G.L., Parallel Implementations of Gradient Based Iterative Algorithms for a Class of Discrete Optimal Control Problems, *1987 International Conference on Parallel Processing,* August, 1987, pp. 491-494.

[POL71] Polak, E.J., *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, New York, 1971.

[SCH81] Scheel, C. and McInnis, B., Parallel Processing of Optimal Control Problems by Dynamic Programming, *Information Sciences,* Vol. 25, No. 2, November 1981, pp. 85-114.

[TRA80] Travassos, R. and Kaufman, H., Parallel Algorithms for Solving Nonlinear Two-Point Boundary-Value Problems Which Arise in Optimal Control, *Journal of Optimization Theory and Applications,* Vol. 30, No. 1, January 1980, pp. 53-71.

Conjugate Points in Optimal Control

J. KOGAN

Department of Mathematics, Purdue University, W. Lafayette, IN 47907

1. THE PROBLEM

We consider a linear control system in $\mathbb{R}^n$:

$$\frac{dx(t)}{dt} = A(t)x(t) + B(t)u(t) \tag{1}$$

with a quadratic cost functional

$$J(x(\cdot),u(\cdot),t_-,t_+) = \frac{1}{2}\int_{t_-}^{t_+} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^* \begin{bmatrix} R_{11}(t) & R_{12}(t) \\ R_{21}(t) & R_{22}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt, \tag{2}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $A(t)$, $B(t)$, $R_{ij}(t)$, $1 \le i,j \le 2$, are matrices with appropriate dimensionalities whose entries are bounded and measurable functions in $(-\infty,\infty)$. The matrix $R_{22}(t)$ is positive definite in $(-\infty,\infty)$.

Definition 1. Let $x(t)$ be an absolutely continuous function from $[t_-,t_+]$ to $\mathbb{R}^n$ and $u(t)$ be a measurable function from $[t_-,t_+]$ to $\mathbb{R}^m$. The pair $(x(t),u(t))$ is admissible if it satisfies (1).

Definition 2. We will say that an admissible pair $(x(t),u(t))$ is an optimal pair on $[t_-,t_+]$ if $J(y(\cdot),w(\cdot),t_-,t_+) \ge J(x(\cdot),u(\cdot),t_-,t_+)$ for all admissible pairs $(y(t),w(t))$ with the same boundary conditions $x(t_-) = y(t_-)$, $x(t_+) = y(t_+)$.

We will concentrate on the following questions: If $x(t)$ is an extremal trajectory (i.e. a curve which satisfy the Maximum Principle) defined on an interval $[t_-,\infty)$ then,

(a) Find a point t_+^O such that $x(t)$ is optimal on the interval $[t_-,t_+^O]$ and not optimal beyond t_+^O.

(b) Find a maximal point t_+^R in $[t_-,t_+^O]$ in (a) such that $x(t)$ is the *unique* optimal trajectory which connects $x(t_-)$ to $x(t_+)$ for any t_+ with $t_- < t_+ < t_+^R$.

(c) Characterize points t_+^R and t_+^O.

2. THE EXAMPLE

Minimize $\frac{1}{2}\int_{t_-}^{t_+}(u^2 - x^2)dt$ over the curves $\dot{x}(t) = b(t)u(t)$ with the boundary conditions $x(t_-) = x(t_+) = 0$, where x, u are scalars, and $b(t)$ is a real valued function given by the relations: $b(t) = 1$ in $(-\infty,-\pi] \cup [0,\pi) \cup (2\pi,\infty)$ and $b(t) = 0$ in $(-\pi,0) \cup [\pi,2\pi]$. Let $t_- = -\pi$. The curves $x_1(t)$ and $x_2(t)$ which are given by the formulas:

$$x_1(t) = 0 \text{ in } [-\pi, \infty),$$

$$x_2(t) = 0 \text{ in } [-\pi, 0) \cup [\pi, 2\pi), \; x_2(t) = \sin t \text{ in } [0, \pi), \; x_2(t) = -\sin t \text{ in } [2\pi, \infty),$$

are extremals. The corresponding costs coincide at $t_+ = \pi$, and $t_+^R = \pi$. For $t_+ > \pi$ the cost along $x_2(t)$ is still optimal until $t_+ = 2\pi$, beyond $t_+ = 2\pi$ both trajectores $x_1(t)$ and $x_2(t)$ stop being optimal, hence $t_+^O = 2\pi$. The scalar Riccati differential equation which corresponds to the problem (1), (2) is $\dot{w}(t) + b^2(t)w^2(t) + 1 = 0$ (see [3], p. 11). The general solution of this equation in $[0, \pi]$ is given by the formula $w(t) = tan(-t + c)$. Hence there exists no solution for this equation on intervals $[-\pi, t_+]$, where $t_+ \geq \pi$. On the other hand the extremal $x_1(t)$ is optimal on $[-\pi, 2\pi]$. It is therefore hopeless to investigate the optimality of extremals beyond π by means of the Riccati equation.

3. THE RESULTS

Theorem 1. (Jacobi condition I.) The point t_+^R is the minimal time beyond t_- such that there exists a nonzero extremal $x(t)$ which starts at 0 at time t_- and returns to 0 at time t_+^R. Solutions of the corresponding Riccati equation blow up at (or before) t_+^R.

Theorem 2. (Jacobi condition II.) The point t_+^O is the minimal point beyond t_- with the following property: There exists an extremal $x(t)$ with $x(t_-) = x(t_+) = 0$ such that the "broken" curve $\hat{x}(t)$ (which is defined by the relations $\hat{x}(t) = x(t)$ in $[t_-, t_+^O]$ and $\hat{x}(t) = 0$ in (t_+^O, ∞)) stops being extremal beyond t_+^O. (This property of t_+^O coinsides with the definition of a conjugate point proposed in [1].)

Theorem 3. If the matrices $A(t)$ and $B(t)$ are such that their entries are analytic functions of t then $t_+^R = t_+^O$. The particular case being the autonomous systems.

REFERENCES

[1] M.R. HESTENES, *On Quadratic Control Problems*, Calculus of Variations and Control Theory **36** (1978), 289–304, Mathematics Research Center of the University of Wisconsin and Academic Press.

[2] V. JURDJEVIC AND J. KOGAN, *Optimality of Extremals for Linear Systems with Quadratic Costs*, Technical Report # 66/May 1987. C.A.M., Purdue University.

[3] W.T. REID, "Riccati Differential Equation," Academic Press, New York, 1972.

TIME SENSITIVITY ANALYSIS OF NONCONVENTIONAL SAMPLED-DATA SYSTEMS

P. Albertos and J. Tornero
Dept. Ing. Sistemas Computadores y Automatica
Univ. Politecnica de Valencia
Spain

INTRODUCTION

With the extended use of computers as controllers, the study of complex sampled-data systems with some continuous and discrete-time components interconnected by means of samplers and hold circuits working at possibly different sampling rates and without synchronicity must be undertaken. Some attempts have been made either with classical tools, [1], - fictitious samplers and z-modified transform - or state variables, [2] and periodic systems [3]. Recently, [4], the authors have presented an approach to this problem splitting any complex system into two parts: components and structure, and events. In this paper this approach will be shown as especially well suited for time sensitivity analysis. Of course, any approach to this problem implies some burden in notation and formulas. So we will restrict ourselves to present the concepts and some illustrative results.

OUTLINE OF THE METHODOLOGY

Four kinds of elements should be considered: continuous-time components, discrete-time components, samplers and hold circuits. In order to cover all possible structures, an extended system vector is defined. Components state variables, outputs of discrete elements as well as external inputs and outputs are included in this vector. Depending upon the sequence of events considered, some of these variables will be removed subsequently and the remaining variables will form the system state vector. The system equations, those internal to the components as well as their interaction, can be stated without timing considerations.

Then, events are considered. These can be discrete-time and continuous-time. The discrete ones cause an instantaneous change in the state vector and determine limits for the continuous ones. Events are arranged in sequence and the sequence is repeated every T second. T is the global system period. This is a constraint but it makes sense in real systems. With these premises, a general evolution matrix can be stated. The equation (1) shows the evolution of the general vector if, after a discrete event, a time interval τ lapses and then all the possible discrete events take place simultaneously. The notation is the usual (for a complete description see, for instance [5]). The zero-elements show a noninteraction and the x-elements a possible direct connection. Each row represents the corresponding variable transition when a related event takes place. If an event does not affect this variable it should remain unchanged. So, for any particular event a transition matrix is constructed with the involved rows of the general evolution matrix, the remaining rows being null except their diagonal element substituted by the identity.

It must be observed that all these transition matrices are time independent except those related to continuous time events, that is, the evolution of the state variables attached to continuous-time components. Elements of this matrix can be parameterized with respect to lapsed time, τ, or computed by self multiplication, given its value for a unitary time interval.

Given a sequence of events, the transition matrices are multiplied in the order determined by this sequence and a transition matrix over the full period T can be computed for control design purposes or just for systems dynamic analysis.

The on-line evaluation of partial transitions according to the sequence of events is a powerful tool for systems simulation.

[†] The author is a Visiting Professor with the Department of Electrical and Computer Engineering and a Visiting Research Professor with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. This visit was supported in part by "Conselleria de C.E. y C. de la Comunidad Autonoma Valenciana," Grant 87/1617.

TIME SENSITIVITY ANALYSIS

The separated treatment of structure, usually fixed during the time considered, and events usually pre-scheduled but subjected to unpredictable disturbances, allows an easy study of the influence of these disturbances in the overall system behavior.

Really, at the simulation level, the system is completely decomposed into elemental items, both components and events, so any possible disturbance can be introduced. For example, changes in some input, loss of some interaction or, what is our interest here, changes in the sequence of events. A computational delay between sampling and the issue of control action can be studied just delaying these events. A loss of synchronicity in the acquisition of data is also studied splitting the sampling event in a set of singular sampler actions. The increase or decrease of the sampling period of a sampler, assuming the constraint of global periodicity, equally means a change in the event sequence.

Any change in the sequence of discrete events represents, at least, two changes in the continuous-time event as a result of the limits shift. Let us denote by $\phi(t_1)$ and $\Gamma(t_1)$ the transition matrix and the input matrix, respectively, for a unitary time interval t_1, corresponding to a continuous-time element. If a time interval $\tau = \lambda t$ (λ integer) is considered, then $\phi(\tau) = \phi^{\lambda}(t_1)$ and (2). These expressions are computed from the transition matrices defined by (1).

For simulation purposes we will adopt as time measurement unit the integration interval. So any continuous-time event spends a time multiple of t_1 and any discrete event may happen at any integration period.

Figure 1 shows some results of a simple system [5] with delay.

REFERENCES

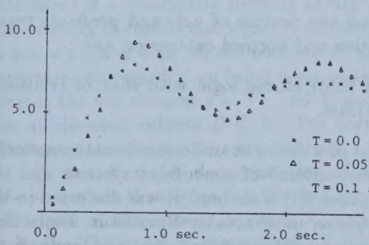
[1] Jury, E. J., "A general z-transform formula for sampled-data systems," *IEEE Trans. Automat. Contr.*, Vol. AC-12, pp. 606-608, 1967.

[2] Kalman, R. E. and Bertram, J. E., "A unified approach to the theory of sampling systems," *Journal of the Franklin Institute*, Vol. 267, pp. 405-436, 1959.

[3] Ackermann, J., *Sampled-Data Control Systems*, Springer-Verlag, pp. 161-170, 1985.

[4] Tornero, J. and Albertos, P., "A new methodology for modelling, analysis and control design of nonconventional sampled data systems," *4th IFAC/IFIP Symp. on Software for Computer Control*, Graz, 1986.

[5] Tornero, J., "Una nueva metodologia para el estudio de sistemas muestreados multifrecuencia," (Spanish), Ph.D. Thesis, Universidad Politecnia de Valencia, September 1985.



$$\begin{bmatrix} x_c \\ x_d \\ x_s \\ x_h \\ y_d \\ u_e \\ y_e \end{bmatrix} = \begin{bmatrix} \phi(\tau) & 0 & 0 & \Gamma(\tau) & 0 & 0 & 0 \\ 0 & x & x & x & x & x & 0 \\ x & 0 & x & x & x & x & 0 \\ 0 & 0 & x & x & x & x & 0 \\ 0 & x & x & x & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & x & x & x & 0 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ x_d \\ x_s \\ x_h \\ y_d \\ u_e \\ y_e \end{bmatrix} \qquad (1)$$

$$\Gamma(\tau) = \sum_{i=0}^{\lambda-1} [\phi^{i}(t_1)\Gamma(t_1)] \qquad (2)$$

INTRODUCTION TO DIALECTICAL NETS

ROBERT E. KENT
Department of Electrical Engineering and Computer Science
University of Illinois at Chicago, Chicago, Illinois

ABSTRACT

This paper initiates the dialectical approach to net theory. This approach views nets as special, but very important and natural, dialectical systems. By following this approach, a suitably generalized version of nets, called *dialectical nets*, can be defined in terms of the "fundamental contradiction" inherent in the structure of *closed preorders*. Dialectical nets are the least conceptual upper bound subsuming the notions of Petri nets, Kan quantification and transition systems. The nature of dialectical nets is that of logical dynamics, and is succinctly defined and summarized in the statement that "dialectical nets are transition systems relativized to closed preorders, and hence are general predicate transformers".

INTRODUCTION

Nets are extensively used to model system phenomena such as concurrency, conflict,synchronization, infomation flow etc. In order to model a variety of systems, nets come in a variety of forms including: condition/event nets using boolean values, consumption/production nets using natural numbers, predicate/transition nets using colored tokens and formal polynomials, etc. In order to make mathematical sense out of this multiplicity of net models, and in order to be able to extend the net concept to first order and higher order logic, we stress the need for a proper mathematical base. The first step in the development of this base is recognition of the fact that values in nets (be they booleans, numbers,colored tokens, subsets, etc.) form a well-known mathematical structure called a *closed preorder*. This is the established and accepted structure for predicates in first order and higher order logic and should also be used in net theory. The second step in the development of this base is recognition of the fact that transitions in nets (be they precondition/postcondition, consumption/production, conjunction/implication, existential-quantification/substitution, etc.) are generalized inverse or dialectical activities and form a well-known mathematical structure called an *adjunction*. After development of a proper mathematical base for nets we introduce a generalized model for such system phenomena called *dialectical nets*. Dialectical nets are special, but very important, forms of dialectical systems based upon the internal contradiction in closed preorders. The theory of dialectical (motion in) systems has already been applied to four important areas of computer science:

- concurrent systems [2], where it distinguishes the notions of observational equivalence and dialectical motion of transition systems;
- OBJ-like functional programming, where it generalizes the notion of *institutions*, and is based upon the doctrinal diagram associated with algebraic theories;
- generalized Petri net theory [this paper], where it unites the notions of *nets* and *predicate transformers*, and is based upon bimodules, Kan quantification and normed categories; and
- first order logic [3], where it unifies the semantics of Horn clause logic with that of relational databases, and is based upon the notion of model doctrines.

The theory of dialectical systems was originally developed out of a desire to understand mathematically the obvious structural similarities between the "parallel composition" of concurrent systems and the "natural join" of database relations. The dialectical view of nature [1] is ancient: it was discussed in the earliest history of ideas by various Presocratic Greek philosophers, most notably Heraclitus. Dialectical systems contain the following essential aspects: 1. based upon contradictions or "opposing tendencies"; 2. interacting objects or entities; 3. movement, motion or development; and 4. reproduction or renewal of entities. All of these aspects are present in the parallel composition of concurrent interacting systems and the natural join of database relations. They are also present as basic concepts in the theory of nets. F.W. Lawvere gave the theory of dialectical systems its most succinct expression: CATEGORY THEORY $\equiv$ OBJECTIVE DIALECTICS. Indeed dialectics invests the dynamical view of systems theory with the fundamental ideas of category theory, such as adjunctions, limits, tensors and Kan extensions; but in turn, it gives these categorical notions that dynamical view. In short, the theory of dialectics studies both the "motion (development, or growth) of structure" and the "structure of motion".

ENRICHED NETS

In the theory of dialectics inverse activities such as consumption and production are fundamental structural units called opposing tendencies, or contradictions. A mathematical formulation of dialectics exists, and is called category theory: CATEGORY THEORY $\equiv$ OBJECTIVE DIALECTICS. In category theory dialectical contradictions are represented by adjunctions. Given two monotonic functions $\mathcal{B} \xrightarrow{f} \mathcal{A}$ and $\mathcal{B} \xleftarrow{u} \mathcal{A}$ flowing in opposite directions between two preorders $\mathcal{A} = \langle A, \preceq_A \rangle$ and $\mathcal{B} = \langle B, \preceq_B \rangle$, the pair $\langle f, u \rangle$ is called an *adjunction* (or an adjoint pair, or a Galois connection, or generalized inverses, or opposing tendencies, or a dialectical contradiction), when they satisfy the equivalence axiom: $f(b) \preceq_A a$ iff $b \preceq_B u(a)$. In this case, f is called the left adjoint (or left aspect) and u is called the right adjoint (or right aspect). The equivalence axiom can be interpreted as the "dialectical tension" which exists between the left and right aspects within the complementary pair. We symbolize this adjunction by the functional notation $(f \dashv u) : \mathcal{B} \longrightarrow \mathcal{A}$ with $\mathcal{B}$ (arbitrarily) the source of the adjunction and $\mathcal{A}$ the target. The following two conditions are equivalent: (1) $(f \dashv u) : \mathcal{B} \longrightarrow \mathcal{A}$; (2) unit axiom $\mathrm{Id}_B \preceq f \cdot u$; that is, $b \preceq_B u(f(b))$ for all $b \in B$; and counit axiom $u \cdot f \preceq \mathrm{Id}_A$; that is, $f(u(a)) \preceq_A a$ for all $a \in A$. Either of these equivalent conditions implies the condition: (3) $u \cdot f \cdot u \equiv u$; and $f \cdot u \cdot f \equiv f$. Ordinary inverses are generalized inverses (contradictions): two monotonic functions $\mathcal{B} \xrightarrow{f} \mathcal{A}$ and $\mathcal{B} \xleftarrow{f^{-1}} \mathcal{A}$ which are inverse to each other, $f \cdot f^{-1} = \mathrm{Id}_B$ and $f^{-1} \cdot f = \mathrm{Id}_A$, form an adjunction $(f \dashv f^{-1}) : \mathcal{B} \longrightarrow \mathcal{A}$. The notion of adjoint pairs can be generalized from the realm of preorders and monotonic functions to the realm of categories and functors.

The fundamental algebraic structure used to define the dynamics of consumption/production Petri nets is that of the natural numbers $\mathbf{N}$. Natural numbers represent quantities of various resources in systems, which are distributed over, and indexed by, places. Certain properties of natural numbers are essential in the definition of the structure and behavior of nets. These properties form a coherent and very important mathematical structure called a closed preorder. A *closed preorder* [5] $\mathbf{V} = \langle V, \preceq, \oplus, \Rightarrow, e \rangle$ consist of the following data and axioms: (1) $\langle V, \preceq, \oplus, e \rangle$ is a monoidal preorder, or ordered monoid, with $\langle V, \preceq \rangle$ a preorder and $\langle V, \oplus, e \rangle$ a monoid, where the binary operation $\oplus : V \times V \longrightarrow V$, called $\mathbf{V}$-composition, is monotonic: if both $u \preceq u'$ and $v \preceq v'$ then $(u \oplus v) \preceq (u' \oplus v')$; (2) $\oplus$ is symmetric, or commutative; that is, $a \oplus b = b \oplus a$ for all elements $a, b \in V$; and (3) $\mathbf{V}$ satisfies the closure axiom: the monotonic $\mathbf{V}$-composition function $(\,) \oplus b : V \longrightarrow V$ has a specified right adjoint $b \Rightarrow (\,) : V \longrightarrow V$ for each element $b \in B$, called $\mathbf{V}$-implication, or symbolically $((\,) \oplus b) \dashv (b \Rightarrow (\,)) : V \longrightarrow V$; that is, $a \oplus b \preceq c$ iff $a \preceq b \Rightarrow c$ for any triple of elements $a, b, c \in V$. The adjunction in the closure axiom is what we referred to as the fundamental contradiction inherent in the mathematical structure of the closed preorder $\mathbf{V}$. The counit axiom for the closure adjunction is generalized *modus ponens*: $((b \Rightarrow a) \oplus b) \preceq a$ for all elements $a, b \in V$. When the unit axiom of the closure adjunction is equivalence, $((b \Rightarrow (a \oplus b)) \equiv a$ for all elements $a, b \in V$, the closed preorder $\mathbf{V}$ is said to be *coreflective*. The commutative, associative and unital binary operation $\oplus$ is sometimes called a tensor product. We usually also assume that our closed preorders are bicomplete; that is, the supremum $\bigvee B$ and the infimum $\bigwedge B$ exist (and are unique up to equivalence $\equiv$) for all subsets $B \subseteq V$. When the tensor product $\oplus$ is the binary infimum or meet $\wedge$ and the unit e is the top element $\top_V$, the closed preorder $\mathbf{V} = \langle V, \preceq, \wedge, \Rightarrow, \top_V \rangle$ is called a *cartesian closed preorder* (or a bicomplete Heyting prealgebra, or a locale). The context of cartesian closed preorders is the context of traditional logic. A characteristic property of cartesian closed preorders is idempotency: $v \oplus v = v \wedge v = v$ for all elements $v \in V$. In a cartesian closed preorder, and even in an arbitrary closed preorder, we regard V as being a set of generalized truth values. A closed preorder is normal when the unit is the top element $e = \top_V$ and $\mathbf{V}$-implication is directed-continuous: $b \Rightarrow (\bigvee_{d \in D} d) \equiv \bigvee_{d \in D} (b \Rightarrow d)$ for all directed subsets $D \subseteq V$. For normal closed preorders $a \oplus b \preceq a \wedge b$ for all elements $a, b \in V$. Cartesian closed preorders are normal.

A pair $\mathcal{X} = \langle X, d_X \rangle$ consisting of a set X and a function $d_X : X \times X \longrightarrow V$ is called a *quasi* $\mathbf{V}$-*space* when it satisfies the triangle (or transitivity) axiom $d_X(x_1, x_2) \oplus d_X(x_2, x_3) \preceq d_X(x_1, x_3)$ for all triples of elements $x_1, x_2, x_3 \in X$; and the zero (or reflexivity) axiom $e \preceq d_X(x, x)$ for all elements $x \in X$. The quasi $\mathbf{V}$-space $\mathcal{X} = \langle X, d_X \rangle$ is a $\mathbf{V}$-space when it satisfies the additional condition: if $e \preceq d_X(x_1, x_2)$ and $e \preceq d_X(x_2, x_1)$ then $x_1 = x_2$. The function d_X is called a metric. We interpret d_X to be either a generalized distance function or a fuzzy preorder. In general our metrics are asymmetrical: $d_X(x_1, x_2) \neq d_X(x_2, x_1)$. Any quasi $\mathbf{V}$-space $\mathcal{X} = \langle X, d_X \rangle$ can be symmetrized by defining $d_X^{\mathrm{sym}}(x_1, x_2) = d_X(x_1, x_2) \oplus d_X^{\mathrm{op}}(x_1, x_2)$ where $d_X^{\mathrm{op}}(x_1, x_2) = d_X(x_2, x_1)$ is the dual or opposite metric. The set of truth values $\mathcal{V} = \langle V, d_V \rangle$, where $d_V(v_1, v_2) = v_1 \Rightarrow v_2$, is a quasi $\mathbf{V}$-space. Any set X can be viewed as a discrete $\mathbf{V}$-space $X = \langle X, d_X \rangle = X^{\mathrm{op}}$, where $d_X(x, x') = e$ if $x = x'$, $= \perp_V$ if $x \neq x'$. Associated with every quasi $\mathbf{V}$-space $\mathcal{X} = \langle X, d_X \rangle$ is an underlying preorder $\Box\mathcal{X} = \langle X, \preceq_X \rangle$ where $x \preceq_X x'$ when $e \preceq d_X(x, x')$, and

x and x' are unrelated when $e \not\preceq d_X(x, x')$. So the characteristic monotonic function for the order $\preceq_X$ is $\kappa_{\preceq_X} = d_X \cdot \Box_V : \langle X, \preceq_X\rangle^{\mathrm{op}} \times \langle X, \preceq_X\rangle \rightarrow \langle V, \preceq\rangle \rightarrow \langle 2, \leq\rangle = \{0 \leq 1\}$, where $\Box_V = e \preceq (\,)$ is the usual characteristic function for the principal filter $\uparrow_V(e) \subseteq V$; that is, $\Box_V(v) = 1$ if $e \preceq v$, and $\Box_V(v) = 0$ otherwise. Note that $\Box(\mathcal{X}^{\mathrm{sym}}) = (\Box\mathcal{X})^{\mathrm{sym}} = \langle X, \equiv_X\rangle$ and that $\Box(\mathcal{X}^{\mathrm{op}}) = (\Box\mathcal{X})^{\mathrm{op}} = \langle X, \succeq_X\rangle$. For a $\mathbf{V}$-space the underlying preorder is a partial order. For a symmetric quasimetric $\mathbf{V}$-space the underlying preorder is an equivalence relation. For the space of generalized truth values $\mathcal{V} = \langle V, d_V\rangle$, since $e \preceq d_V(v_1, v_2)$ iff $v_1 \preceq v_2$, the underlying preorder is the given order on $\mathbf{V}$.

A $\mathbf{V}$-*morphism* $f : \mathcal{X} \longrightarrow \mathcal{Y}$ between two quasi $\mathbf{V}$-spaces $\mathcal{X} = \langle X, d_X\rangle$ and $\mathcal{Y} = \langle Y, d_Y\rangle$ is a function $f : X \longrightarrow Y$ which satisfies the condition $d_X(x, x') \preceq d_Y(f(x), f(x'))$ for all $x, x' \in X$. $\mathbf{V}$-spaces and $\mathbf{V}$-morphisms form the category $\mathbf{Space}_V$. By modus ponens, $(\,) \oplus v : \mathcal{V} \longrightarrow \mathcal{V}$ is a $\mathbf{V}$-morphism for all elements $v \in V$. By transitivity of d_V, $v \Rightarrow (\,) : \mathcal{V} \longrightarrow \mathcal{V}$ is a $\mathbf{V}$-morphism for all elements $v \in V$. Given any two quasi $\mathbf{V}$-spaces $\mathcal{X}$ and $\mathcal{Y}$ the set of all $\mathbf{V}$-morphisms from $\mathcal{X}$ to $\mathcal{Y}$ is a quasi $\mathbf{V}$-space $\mathcal{Y}^{\mathcal{X}}$, called the exponential quasi $\mathbf{V}$-space of $\mathcal{X}$ and $\mathcal{Y}$, whose metric d, called the pointwise inf metric, is defined by $d(f, g) = \bigwedge_{x \in X} d_Y(f(x), g(x))$. Notice that the metric d_X is not used to define d. The metric d_X is only used to restrict admission to the underlying set of $\mathcal{Y}^{\mathcal{X}}$. In particular, the exponential space $\mathcal{V}^{\mathcal{X}}$ of all $\mathbf{V}$-valued $\mathbf{V}$-morphisms on $\mathcal{X}$ is an quasi $\mathbf{V}$-space with the inf metric $d(\phi, \psi) = \bigwedge_{x \in X} d_V(\phi(x), \psi(x)) = \bigwedge_{x \in X}[\phi(x) \Rightarrow \psi(x)]$. We interpret an element of $\mathcal{V}^{\mathcal{X}}$, a $\mathbf{V}$-morphism $\mu : \mathcal{X} \longrightarrow \mathcal{V}$, to be an X-indexed marking $\mu : X \longrightarrow \mathcal{V}$ which satisfies the internal pointwise metric constraint d_X: $d_X(x, x') \preceq d_V(\mu(x), \mu(x')) = \mu(x) \Rightarrow \mu(x')$ for all $x, x' \in X$; or equivalently, by the $\oplus$-$\Rightarrow$ adjunction, $\mu(x) \oplus d_X(x, x') \preceq \mu(x')$ for all $x, x' \in X$. Such a marking $\mu : \mathcal{X} \rightarrow \mathbf{V}$ which is constrained by the metric d_X is called a $\mathbf{V}$-(*valued*) *predicate* over $\mathcal{X}$: "predicates $\equiv$ metric-constrained markings". The specification $\perp_V = d_X(x, x')$ is no constraint at all, and that the specification $e \preceq d_X(x, x')$ is precisely the order-theoretic constraint $x \preceq x'$ requiring that μ satisfy $\mu(x) \preceq \mu(x')$. For the exponential space $\mathcal{V}^{\mathcal{X}}$ of $\mathbf{V}$-predicates over $\mathcal{X}$, since $e \preceq d(\phi, \psi)$ iff $e \preceq \bigwedge_{x \in X} d_V(\phi(x), \psi(x))$ iff $e \preceq d_V(\phi(x), \psi(x))$ for all $x \in X$ iff $\phi(x) \preceq \psi(x)$ for all $x \in X$, the underlying preorder is the usual *entailment order* on $\mathbf{V}$-predicates over $\mathcal{X}$.

We relativize the notion of a consumption/production net by using as our fundamental domain of values an arbitrary closed preorder $\mathbf{V}$ in place of the natural numbers $\mathbf{N}$. A $\mathbf{V}$-*net* N is a quadruple $\mathsf{N} = \langle T, \mathcal{P}, \iota, o\rangle$ consisting of: a set (of transition symbols) T, a quasi $\mathbf{V}$-space (of places) $\mathcal{P}$, an input weighting function $\iota : T \longrightarrow \mathbf{V}^{\mathcal{P}}$, and an output weighting function $o : T \longrightarrow \mathbf{V}^{\mathcal{P}}$. Markings are given two interpretations: (1) a place p is a site associated with a value (of a resource) $\mu(p)$ and a marking μ is a distribution of places (hence resources); or (2) a marking μ is a fuzzy P-subset with $\mu(p)$ indicating the degree-of-membership of "$p \in \mu$". A transition $t \in T$ in a $\mathbf{V}$-net N is enabled by marking μ when $\mu \preceq \iota(t)$; that is, when $\mu(p) \preceq \iota(t, p)$ for all places $p \in P$. A transition $t \in T$ fires by $\Rightarrow$-ing, or consuming, $\iota(t, p)$ tokens from place $p \in P$, and then $\oplus$-ing, or producing, $o(t, p)$ tokens to place $p \in P$. The result of the firing of a transition $t \in T$ is expressed by the equation $\mathsf{N}_t(\mu)(p) = [\iota(t, p) \Rightarrow \mu(p)] \oplus o(t, p)$ for all places $p \in P$. We regard a $\mathbf{V}$-net to be a transformer of constrained $\mathbf{V}$-markings; that is, a $\mathbf{V}$-predicate transformer. The semantics of a $\mathbf{V}$-net N can be defined as either external or internal behaviors. External behaviors include: (1) unfoldment-tree, and (2) regular-set behavior. Internal behaviors include: (1) reachable predicates (markings), and (2) cumulative fixpoint behavior.

We list some important closed preorders on which nets and transition systems can be based:

booleans [cartesian closed]

$\mathbf{2} = \langle 2 = \{0, 1\}, \leq, \wedge, \rightarrow, 1\rangle$, where 0 is **false**, 1 is **true**, $\leq$ is the usual order on truth-values, $\wedge$ is the truth-table for **and**, and $\rightarrow$ is the truth-table for **implies**. Here quasi $\mathbf{2}$-spaces $\mathcal{X} = \langle X, d\rangle$ are preorders $\mathcal{X} = \langle X, \preceq\rangle$ where $x_1 \preceq x_2$ when $d(x_1, x_2) = 1$, $\mathbf{2}$-spaces are posets, and $\mathbf{2}$-morphisms are monotonic functions.

$\mathbf{2}' = \langle 2 = \{1, 0\}, \geq, \vee, \setminus, 0\rangle$, where 1 is **true**, 0 is **false**, $\geq$ is the usual downward order on truth-values, $\vee$ is the truth-table for **or**, and $\setminus$ is the truth-table for **difference**: $b_1 \setminus b_2$ is true iff b_1 is true and b_2 is false. Here, quasi $\mathbf{2}'$-spaces are preorders where $x_1 \preceq x_2$ when $d(x_1, x_2) = 0$, $\mathbf{2}'$-spaces are posets, and $\mathbf{2}'$-morphisms are monotonic functions. $\mathbf{2}'$ defines the correct context for condition/event nets.

natural numbers

$\mathbf{N} = \langle N, \geq, +, \dot{-}, 0\rangle$, where N is the set of natural numbers $N = \{0, 1, \ldots, n, \ldots, \infty\}$ with infinity, $\geq$ is the usual downward ordering on natural numbers N, $+$ is sum, and $\dot{-}$ is difference defined by $m \dot{-} n = m - n$ if $m \geq n$, $= 0$ if $m < n$.

reals

$\mathbf{R} = \langle R = [\infty, 0], \geq, +, \dot{-}, 0\rangle$.

The quantitative closed preorders of reals $\mathbf{R}$ and natural numbers $\mathbf{N}$ are coreflective and normal. They define the correct context for consumption/production nets.

markings
If $\mathbf{V}$ is a closed preorder and I is any indexing set, then the marking space $\mathcal{V}^I$ is a closed preorder $\mathbf{V}^I = \langle V^I, \preceq, \oplus, \Rightarrow, e \rangle$ where $\preceq, \oplus, \Rightarrow$ and e have obvious pointwise definitions. I might denote places, colors, some combination of these, etc., in nets. If $\mathbf{V} = \langle V, \preceq, \wedge, \Rightarrow, \top_V \rangle$ is a cartesian closed preorder and $\mathcal{X}$ is any $\mathbf{V}$-space, then the predicate space $\mathbf{V}^{\mathcal{X}}$, the restriction of $\mathbf{V}^X$ to $\mathbf{V}$-morphisms, is a [cartesian closed] subpreorder of $\mathbf{V}^X$.
subsets [cartesian closed]
Let A be any set and let $P(A)$ be the set $P(A) = \{B \mid B \subseteq A\}$ of all subsets of A.
$\mathbf{P}(A) = \langle P(A), \subseteq, \cap, \rightarrow, A \rangle$, where $\cap$ is **set intersection**, and $\rightarrow$ is **set implication**: $B_1 \rightarrow B_2 = \{a \in A \mid a \in B_1 \text{ implies } a \in B_2\} = -B_1 \cup B_2$. $\mathbf{P}(A)$ is essentially the marking space closed preorder $\mathbf{P}(A) \cong \mathbf{2}^A$ defining the most basic markings-as-fuzzy-subsets interpretation for nets.
$\mathbf{P}'(A) = \langle P(A), \supseteq, \cup, \setminus, \emptyset \rangle$, where $\cup$ is **set union**, and $\setminus$ is **set difference**: $B_1 \setminus B_2 = \{a \in A \mid a \in B_1 \text{ but not } a \in B_2\} = B_1 \cap -B_2$. $\mathbf{P}'(A)$ is essentially the marking space closed preorder $\mathbf{P}'(A) \cong \mathbf{2}'^A$, the marking space for condition/event nets.

There is a standard method for transforming between enriched contexts. A closed monotonic function $h : \mathbf{V} \longrightarrow \mathbf{W}$ between two closed preorders $\mathbf{V}$ and $\mathbf{W}$ is a monotonic function $h : \mathcal{V} \longrightarrow \mathcal{W}$ which preserves monoidal unit and composition: $e_{\mathbf{W}} \preceq_{\mathbf{W}} h(e_{\mathbf{V}})$ and $h(v) \oplus_{\mathbf{W}} h(v') \preceq_{\mathbf{W}} h(v \oplus_{\mathbf{V}} v')$ for all elements $v, v' \in V$. Closed monotonic functions determine transformations of spaces, predicates, relations, dialectical moves, dialectical nets, etc. For example, any closed preorder $\mathbf{V}$ has a canonical closed monotonic function $\Box_V : \mathbf{V} \longrightarrow \mathbf{2}$, defined above, which determines a functor $\Box_V : \mathbf{Space}_V \longrightarrow \mathbf{Space}_2 = \mathbf{PO}$, where $\Box_V(\mathcal{X}) = \langle X, \preceq_X \rangle$ the underlying preorder of $\mathcal{X}$. Some simple net oriented examples: (1) the two monotonic functions, inclusion $\mathrm{Inc} : \mathbf{N} \longrightarrow \mathbf{Z}$ and saturation $[\,] : \mathbf{Z} \longrightarrow \mathbf{N}$, between natural numbers $\mathbf{N}$ and integers $\mathbf{Z}$, where $[n] = n$ if $n \geq 0, = 0$ if $n < 0$, are adjoint $\mathrm{Inc} \dashv [\,]$ and closed (allowing the use of linear algebraic techniques in net theory); (2) the two monotonic functions, floor $\lfloor\,\rfloor : \mathbf{R} \longrightarrow \mathbf{N}$ and inclusion $\mathrm{Inc} : \mathbf{N} \longrightarrow \mathbf{R}$, between (nonnegative) reals $\mathbf{R}$ and natural numbers $\mathbf{N}$, are adjoint $\lfloor\,\rfloor \dashv \mathrm{Inc}$ and closed (encouraging the use of analysis techniques in net theory).

If $\mathbf{V} = \langle V, \preceq, \oplus, \Rightarrow, e \rangle$ is not a cartesian closed preorder, such as $\mathbf{N}$, then $\mathbf{V}^{\mathcal{X}}$ is not necessarily a closed preorder since it may not be closed under the pointwise operations $\oplus$ and $\Rightarrow$. In one sense the problem with the pointwise and discrete operations of implication $\theta \Rightarrow (\,)$ and composition $(\,) \oplus \theta$ is that they are "isolated" notions without "collective" influence between points, whereas the metric d makes X into a nondiscrete structure $\mathcal{X} = \langle X, d \rangle$ with points which are not isolated from one another in the sense that they have collective constraints between themselves. One general solution to this problem is the use of enriched relations to model the dialectical movement of consumption and production. Relations allow for collective influence between points.

DIALECTICAL NETS

Each element $x \in X$ of a quasi $\mathbf{V}$-space $\mathcal{X} = \langle X, d \rangle$ can be represented as the $\mathbf{V}$-predicate $y(x) = d(x, -)$ over $\mathcal{X}$ where $y(x)(x') = d(x, x')$ for each element $x' \in X$. The map $y : X \longrightarrow V^X$, which is called the *Yoneda embedding*, is a $\mathbf{V}$-isometry $\mathrm{y}_{\mathcal{X}} : \mathcal{X}^{\mathrm{op}} \longrightarrow \mathbf{V}^{\mathcal{X}}$. Composition on the right with the Yoneda embedding allows us to consider the concept of a $\mathbf{V}$-morphism $\mathcal{Y}^{\mathrm{op}} \longrightarrow \mathbf{V}^{\mathcal{X}}$ to be a generalization of the concept of a $\mathbf{V}$-morphism $\mathcal{Y} \longrightarrow \mathcal{X}$. Such a generalized $\mathbf{V}$-morphism is equivalent to a $\mathbf{V}$-morphism $\mathcal{Y}^{\mathrm{op}} \times \mathcal{X} \xrightarrow{\tau} \mathbf{V}$ and may be regarded to be a $\mathbf{V}$-*(valued) relation* (also called a $\mathbf{V}$-*bimodule*) from Y to X, denoted by $\mathcal{Y} \xrightarrow{\tau} \mathcal{X}$, with $\tau(y, x)$ a element of $\mathbf{V}$ being interpreted as the "truth-value of the τ-relatedness of y to x" [5]. We can regard a $\mathbf{V}$-relation to be a $|\mathcal{Y}| \times |\mathcal{X}|$-matrix whose (y, x)-th entry is $\tau(y, x)$. As mentioned above every $\mathbf{V}$-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ determines a $\mathbf{V}$-relation $\mathcal{Y} \xrightarrow{f^{\triangleleft}} \mathcal{X}$ defined by $f^{\triangleleft} = f^{\mathrm{op}} \cdot \mathrm{y}_{\mathcal{X}}$; that is, $f^{\triangleleft}(y, x) = d_X(f(y), x)$. Dually every $\mathbf{V}$-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ also determines a $\mathbf{V}$-relation $\mathcal{X} \xrightarrow{f_{\triangleleft}} \mathcal{Y}$ defined by $f_{\triangleleft} = \mathrm{y}_{\mathcal{X}} \cdot \mathbf{V}^f$; that is, $f_{\triangleleft}(x, y) = d_X(x, f(y))$. A pair of $\mathbf{V}$-relations $\mathcal{Z} \xrightarrow{\sigma} \mathcal{Y}$ and $\mathcal{Y} \xrightarrow{\rho} \mathcal{X}$ can be composed, yielding the $\mathbf{V}$-relation $\mathcal{Z} \xrightarrow{\sigma \circ \rho} \mathcal{X}$ defined to be the categorical *coend* $\sigma \circ \rho(z, x) = \int^{y \in \mathcal{Y}} [\sigma(z, y) \oplus \rho(y, x)] = \bigvee_{y \in \mathcal{Y}} [\sigma(z, y) \oplus \rho(y, x)]$ where $\bigvee$ denotes supremum, which is colimit, in $\mathbf{V} = \langle V, \preceq \rangle$. Relational composition can be viewed as a matrix product. One can verify that relational composition is associative $(\tau \circ \sigma) \circ \rho = \tau \circ (\sigma \circ \rho)$, and that metrics (as $\mathbf{V}$-relations) are identities $d_Y \circ \tau = \tau = \tau \circ d_X$. So $\mathbf{V}$-spaces and $\mathbf{V}$-relations form a category $\mathbf{rel}_V$. One can also verify that $(g \cdot f)^{\triangleleft} = g^{\triangleleft} \circ f^{\triangleleft}$ for any two composable $\mathbf{V}$-morphisms $\mathcal{Z} \xrightarrow{g} \mathcal{Y} \xrightarrow{f} \mathcal{X}$, and that $(\mathrm{Id}_{\mathcal{X}})^{\triangleleft} = d_X$ the identity $\mathbf{V}$-relation at $\mathcal{X}$. So the Yoneda embedding determines a functor $(\,)^{\triangleleft} : \mathbf{Space}_V \longrightarrow \mathbf{rel}_V$ which makes concrete the concept generalization discussed at the beginning of this section.

There is a concept orthogonal to relations-as-morphisms. Given any two $\mathbf{V}$-relations $\mathcal{Y}_1 \xrightarrow{\tau_1} \mathcal{X}_1$

and $\mathcal{Y}_2 \overset{\tau_2}{\rightharpoonup} \mathcal{X}_2$ a (*vertical*) *morphism* of **V**-relations $\langle g, f\rangle : \langle \mathcal{Y}_1, \tau_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, \tau_2, \mathcal{X}_2\rangle$ consists of two **V**-morphisms, a source morphism $g : \mathcal{Y}_1 \longrightarrow \mathcal{Y}_2$ and a target morphism $f : \mathcal{X}_1 \longrightarrow \mathcal{X}_2$, which satisfy the inequality condition $\tau_1 \preceq (g^{\mathrm{op}} \times f) \cdot \tau_2$ as **V**-predicates over $\mathcal{Y}_1^{\mathrm{op}} \times \mathcal{X}_1$; or more abstractly, in terms of relational composition, $g_{\triangleleft} \circ \tau_1 \preceq \tau_2 \circ f_{\triangleleft}$. **V**-relations and their vertical morphisms form a category $\mathbf{Rel}_V$. This is actually the vertical category of a double category, which we also denote by $\mathbf{Rel}_V$, whose underlying horizontal category is $\mathbf{rel}_V$. If we change the definition of a vertical morphism $\langle f, g\rangle : \langle \mathcal{Y}_1, \tau_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, \tau_2, \mathcal{X}_2\rangle$ to the inequality condition $g^{\triangleleft} \circ \tau_2 \preceq \tau_1 \circ f^{\triangleleft}$, we can define a dual (vertical) category, which we denote by $\mathbf{Rel}_V^{\bullet}$. For any category **C** the category of parallel pairs of **C**-morphisms, denoted by $\mathbf{C}^{\times}$, has the same objects as **C**, $|\mathbf{C}^{\times}| = |\mathbf{C}|$, and has parallel pairs of **C**-morphism as its morphisms, $\mathbf{C}^{\times}[c, c'] = (\mathbf{C}[c, c'])^2$. $\mathbf{C}^{\times}$ is a kind of 2nd power (square) of **C**. In particular, $\mathbf{rel}_V^{\times}$ is the category of parallel relation pairs $\tau = \mathcal{Y} \overset{o}{\underset{\iota}{\rightrightarrows}} \mathcal{X}$ with (horizontal) $\mathbf{rel}_V$-composition. Then $\mathbf{rel}_V^{\times}$, with the vertical morphisms $\langle g, f\rangle : \langle \mathcal{Y}_1, \tau_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, \tau_2, \mathcal{X}_2\rangle$ when $\langle g, f\rangle : \langle \mathcal{Y}_1, o_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, o_2, \mathcal{X}_2\rangle$ is a vertical morphism in $\mathbf{Rel}_V$ and $\langle f, g\rangle : \langle \mathcal{Y}_1, \iota_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, \iota_2, \mathcal{X}_2\rangle$ is a vertical morphism in $\mathbf{Rel}_V^{\bullet}$, is a vertical category denoted $\mathbf{Rel}_V^{\times}$.

Any element $v \in V$ is a **V**-relation $\mathbf{1} \overset{v}{\rightharpoonup} \mathbf{1}$. In fact, $\mathbf{rel}_V[\mathbf{1}, \mathbf{1}] = \mathbf{V}^{\mathbf{1}^{\mathrm{op}} \times \mathbf{1}} \cong \mathbf{V}$. Any **V**-relation $\mathbf{1} \overset{\psi}{\rightharpoonup} \mathcal{Y}$, a *generalized element* of $\mathcal{Y}$, is a **V**-morphism $\mathbf{1}^{\mathrm{op}} \times \mathcal{Y} \xrightarrow{\psi} \mathbf{V}$, and hence is the same thing as a predicate over $\mathcal{Y}$: "$\mathcal{Y}$-predicates $\equiv$ generalized $\mathcal{Y}$-elements". In fact, $\mathbf{rel}_V[\mathbf{1}, \mathcal{Y}] = \mathbf{V}^{\mathbf{1}^{\mathrm{op}} \times \mathcal{Y}} \cong \mathbf{V}^{\mathcal{Y}}$. So any **V**-relation $\mathcal{Y} \overset{\tau}{\rightharpoonup} \mathcal{X}$ defines by composition a **V**-morphism $\mathbf{V}^{\mathcal{Y}} \xrightarrow{\curvearrowright_\tau} \mathbf{V}^{\mathcal{X}}$ called *direct flow* (or *yang*), which maps **V**-predicates over $\mathcal{Y}$ to **V**-predicates over $\mathcal{X}$, and is defined by $\curvearrowright_\tau(\psi) = \psi \circ \tau$. The direct flow $\curvearrowright_\tau$ corresponds to the direct image map $PY \xrightarrow{R_P} PX$ of a ordinary binary relation $R \subseteq Y \times X$. Explicitly, $\curvearrowright_\tau$ is defined to be the coend $\curvearrowright_\tau(\psi)(x) = \int^{y \in \mathcal{Y}} [\psi(y) \oplus \tau(y, x)] = \bigvee_{y \in \mathcal{Y}} [\psi(y) \oplus \tau(y, x)]$ for any predicate $\psi \in \mathbf{V}^{\mathcal{Y}}$ over $\mathcal{Y}$ and any point $x \in \mathcal{X}$. Note that $\curvearrowright_{\sigma \circ \rho} = \curvearrowright_\sigma \cdot \curvearrowright_\rho$ for any pair of composable **V**-relations $\mathcal{Z} \overset{\sigma}{\rightharpoonup} \mathcal{Y}$ and $\mathcal{Y} \overset{\rho}{\rightharpoonup} \mathcal{X}$, and that $\tau_1 \preceq \tau_2$ implies $\curvearrowright_{\tau_1} \preceq \curvearrowright_{\tau_2}$ as **V**-morphisms for all pairs $\mathcal{Y} \overset{\tau_1}{\underset{\tau_2}{\rightrightarrows}} \mathcal{X}$. For the special case $\mathcal{Y} = \mathbf{1} = \mathcal{X}$, when **V**-elements are viewed as **V**-relations $\mathbf{1} \overset{v}{\rightharpoonup} \mathbf{1}$, direct flow is $\curvearrowright_v = (\,) \oplus v : \mathbf{V} = \mathbf{V}^{\mathbf{1}} \longrightarrow \mathbf{V}^{\mathbf{1}} = \mathbf{V}$ since $\curvearrowright_v(u) = \bigvee_{\mathbf{1}} (u \oplus v) = u \oplus v$. For any **V**-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ with associated **V**-relation $\mathcal{X} \overset{f_{\triangleleft}}{\rightharpoonup} \mathcal{Y}$, note that $\curvearrowright_{f_{\triangleleft}}(\phi) = f \cdot \phi$ for any **V**-predicate $\phi \in \mathbf{V}^{\mathcal{X}}$ over $\mathcal{X}$; that is, $\curvearrowright_{f_{\triangleleft}}$ is just composition on the right $\curvearrowright_{f_{\triangleleft}} = f \cdot (\,) = \mathbf{V}^f$. On the other hand, for any **V**-morphism **V**-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ with associated **V**-relation $\mathcal{Y} \overset{f^{\triangleleft}}{\rightharpoonup} \mathcal{X}$, the **V**-morphism $\exists_f = \curvearrowright_{f^{\triangleleft}} : \mathbf{V}^{\mathcal{Y}} \longrightarrow \mathbf{V}^{\mathcal{X}}$ is called *existential Kan quantification* along f; in detail, $\exists_f(\psi)(x) = \bigvee_{y \in \mathcal{Y}} [\psi(y) \oplus d_X(f(y), x)]$ for any predicate $\psi \in \mathbf{V}^{\mathcal{Y}}$ over $\mathcal{Y}$ and any point $x \in \mathcal{X}$. Applying the directflow operator to the inequality condition for vertical morphisms in $\mathbf{Rel}_V$ gives $\mathbf{V}^g \cdot \curvearrowright_{\tau_1} \preceq \curvearrowright_{\tau_2} \cdot \mathbf{V}^f$.

The **V**-morphism $\curvearrowright_\tau$ has as a right adjoint the **V**-morphism $\mathbf{V}^{\mathcal{Y}} \xleftarrow{\curvearrowleft_\tau} \mathbf{V}^{\mathcal{X}}$ called *inverse flow* (or *yin*), which maps **V**-predicates over $\mathcal{X}$ to **V**-predicates over $\mathcal{Y}$, and is defined to be the categorical *end* $\curvearrowleft_\tau(\phi)(y) = \int_{x \in \mathcal{X}} [\tau(y, x) \Rightarrow \phi(x)] = \bigwedge_{x \in \mathcal{X}} [\tau(y, x) \Rightarrow \phi(x)] = \bigwedge_{x \in \mathcal{X}} d_V(\tau(y, x), \phi(x))$ for any predicate $\phi \in \mathbf{V}^{\mathcal{X}}$ over $\mathcal{X}$ and any point $y \in \mathcal{Y}$, where $\bigwedge$ denotes infimum, which is limit, in $\mathbf{V} = \langle V, \preceq\rangle$. Note that $\curvearrowleft_{\sigma \circ \rho} = \curvearrowleft_\rho \cdot \curvearrowleft_\sigma$ for any pair of composable **V**-relations $\mathcal{Z} \overset{\sigma}{\rightharpoonup} \mathcal{Y}$ and $\mathcal{Y} \overset{\rho}{\rightharpoonup} \mathcal{X}$, and that $\tau_1 \preceq \tau_2$ implies $\curvearrowleft_{\tau_2} \preceq \curvearrowleft_{\tau_1}$ as **V**-morphisms for all pairs $\mathcal{Y} \overset{\tau_1}{\underset{\tau_2}{\rightrightarrows}} \mathcal{X}$. For the special case $\mathcal{Y} = \mathbf{1} = \mathcal{X}$, when **V**-elements are viewed as **V**-relations $\mathbf{1} \overset{v}{\rightharpoonup} \mathbf{1}$, inverse flow is $\curvearrowleft_v = v \Rightarrow (\,) : \mathbf{V} = \mathbf{V}^{\mathbf{1}} \longrightarrow \mathbf{V}^{\mathbf{1}} = \mathbf{V}$ since $\curvearrowleft_v(u) = \bigwedge_{\mathbf{1}} (v \Rightarrow u) = v \Rightarrow u$. For any **V**-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ with associated **V**-relation $\mathcal{X} \overset{f^{\triangleleft}}{\rightharpoonup} \mathcal{Y}$, note that $\curvearrowleft_{f^{\triangleleft}}(\phi) = f \cdot \phi$ for any **V**-predicate $\phi \in \mathbf{V}^{\mathcal{X}}$ over $\mathcal{X}$; that is, $\curvearrowleft_{f^{\triangleleft}}$ is just composition on the right $\curvearrowleft_{f^{\triangleleft}} = f \cdot (\,) = \mathbf{V}^f = \curvearrowright_{f_{\triangleleft}}$. On the other hand for any **V**-morphism $\mathcal{Y} \xrightarrow{f} \mathcal{X}$ with associated **V**-relation $\mathcal{X} \overset{f_{\triangleleft}}{\rightharpoonup} \mathcal{Y}$, the **V**-morphism $\forall_f = \curvearrowleft_{f_{\triangleleft}} : \mathbf{V}^{\mathcal{Y}} \longrightarrow \mathbf{V}^{\mathcal{X}}$ is called *universal Kan quantification* along f; in detail, $\forall_f(\psi)(x) = \bigwedge_{y \in \mathcal{Y}} [d_X(x, f(y)) \Rightarrow \psi(y)]$ for any predicate $\psi \in \mathbf{V}^{\mathcal{Y}}$ over $\mathcal{Y}$ and any point $x \in \mathcal{X}$. Ordinary quantification in predicate calculus is a very special case of Kan quantifiction. Kan quantification is quantification relativized to an arbitrary closed preorder **V**. Applying the inverseflow operator to the inequality condition for vertical morphisms in $\mathbf{Rel}_V^{\bullet}$ gives $\mathbf{V}^f \cdot \curvearrowleft_{\tau_1} \preceq \curvearrowleft_{\tau_2} \cdot \mathbf{V}^g$. The following fact appears in [5], showing that (parallel pairs of) relations specify "the dialectical flow of predicates".

Fact 1 *Direct flow is left adjoint to inverse flow* $\curvearrowright_\tau \dashv \curvearrowleft_\tau$ *for any* **V**-*relation* $\mathcal{Y} \overset{\tau}{\rightharpoonup} \mathcal{X}$.

Dialectical flow is the alternation-composition of direct and inverse flow. A parallel pair $\tau = \mathcal{Y} \overset{o}{\underset{\iota}{\rightrightarrows}} \mathcal{X}$ of **V**-relations is called a *dialectical flow specifier*. Given a flow specifier τ, the *dialectical flow* (or *yinyang*) specified by τ is the composition

$$\circledcirc_\tau = \mathbf{V}^{\mathcal{X}} \xrightarrow{\curvearrowleft_\iota} \mathbf{V}^{\mathcal{Y}} \xrightarrow{\curvearrowright_o} \mathbf{V}^{\mathcal{X}}.$$

The symbol $\circledcirc$ denotes dialectical motion. This is a stylized version of the *yin-yang* symbol of the ancient Chinese *naturalist* philosophers, where the two parts *yin* and *yang* represent contradictory elements or opposing tendencies forming a complementary union out of which all things develop. For us, the complementary union is the flow adjunction and the circular shape represents the cyclical or spiral shape of dialectical motion. Dialectical flow is functorial: if $\langle g, f\rangle : \langle \mathcal{Y}_1, \tau_1, \mathcal{X}_1\rangle \Rightarrow \langle \mathcal{Y}_2, \tau_2, \mathcal{X}_2\rangle$ is a vertical morphism in $\mathbf{Rel}_V^{\times}$, then $\mathbf{V}^f \cdot \circledcirc_{\tau_1} \preceq \circledcirc_{\tau_2} \cdot \mathbf{V}^f$. There is a four-fold duality in the definition of dialectical flow. The *dialectical opflow* (or *yangyin*) specified by τ is the composition $\oslash_\tau = \curvearrowright_o \cdot \curvearrowleft_\iota$. The *dialectical coflow* of τ is the dialectical flow of τ^{op}; namely, $\circledcirc_{\tau^{\mathrm{op}}} = \curvearrowleft_o \cdot \curvearrowright_\iota$. The *dialectical coopflow* of τ is the dialectical opflow of τ^{op}; namely, $\oslash_{\tau^{\mathrm{op}}} = \curvearrowright_\iota \cdot \curvearrowleft_o$.

The modern theory of dialectics incorporates the notion of the *reproduction* of entities which are in dialectical motion. Reproduction can be modelled mathematically as the recursive specification of entities with respect to dialectical motion. Reproduction (renewal, recursion) is the internal semantics of dialectical motion. Dialectical entities in the context of this paper are the enriched predicates (constrained markings). For any flow specifier τ, as above, we say that τ *reproduces* the $\mathbf{V}$-predicate $\phi \in \mathbf{V}^{\mathcal{X}}$ when ϕ is a fixpoint solution of the recursive equation

$$\chi \equiv \circledcirc_\tau(\chi);$$

that is, when $\phi \equiv \circledcirc_\tau(\phi) = \curvearrowright_o(\curvearrowleft_\iota(\phi))$. A fixpoint solution ϕ is an internal behavior of the dialectical motion (specified by) τ. A measure, or index, of reproduction is the value $|\phi|_\tau = \circledcirc_\tau^{\triangleleft}(\phi) = d^{\mathrm{sym}}(\phi, \circledcirc_\tau(\phi))$. So the dialectical flow τ reproduces the entity ($\mathbf{V}$-predicate) ϕ when $e \preceq |\phi|_\tau$. The increasing sequence $\bot_{\mathcal{X}} \preceq \circledcirc_\tau(\bot_{\mathcal{X}}) \preceq \cdots \preceq \circledcirc_\tau^n(\bot_{\mathcal{X}}) \preceq \cdots$ of $\mathbf{V}$-predicates over quasi $\mathbf{V}$-space $\mathcal{X}$, is called the least fixpoint approximation sequence of τ. The $\mathbf{V}$-predicate $\tau^* = \bigvee_{n\in\omega} \circledcirc_\tau^n(\bot_{\mathcal{X}})$, the *least fixpoint* solution of the recursive equation above, exists for directed-continuous inverse flow (yin), and in particular, for normal $\mathbf{V}$. The decreasing sequence $\cdots \preceq \circledcirc_\tau^n(\top_{\mathcal{X}}) \preceq \cdots \preceq \circledcirc_\tau(\top_{\mathcal{X}}) \preceq \top_{\mathcal{X}}$ of $\mathbf{V}$-predicates over quasi $\mathbf{V}$-space $\mathcal{X}$, is called the greatest fixpoint approximation sequence of τ. The $\mathbf{V}$-predicate $\tau^\infty = \bigwedge_{n\in\omega} \circledcirc_\tau^n(\top_{\mathcal{X}})$, is the *greatest fixpoint* solution of the recursive equation above. The least and greatest fixpoints are two canonical internal behaviors of the dialectical motion τ. [A philosophical note: The notion of complementary union (two working together in one) is not that of "synthesis". Neither of the opposites is "transformed". Indeed, with synthesis, dialectical motion would cease! The notion of "reproduction" is one of equilibrium of motion, not lack of motion.]

Just as in the ordinary context of sets a subset $A \subseteq X$ can be viewed as a binary relation $A \subseteq X \times X$ (yAx iff $x \in A$, or $R_A = \mathrm{pr}_2 \cdot \kappa_A$), so also in the context of $\mathbf{V}$-spaces a marking $\theta : X \longrightarrow V$ can be viewed as a $\mathbf{V}$-relation $\mathcal{X} \xrightarrow{\tau_\theta} \tilde{\mathcal{X}}$ for any two quasi $\mathbf{V}$-spaces $\mathcal{X} = \langle X, d\rangle$ and $\tilde{\mathcal{X}} = \langle X, \tilde{d}\rangle$. In this spirit and following the places-as-sites interpretation of nets, for any two quasi $\mathbf{V}$-spaces $\mathcal{X}$ and $\tilde{\mathcal{X}}$, and any $\mathbf{V}$-relation $\mathcal{X} \xrightarrow{\tau} \tilde{\mathcal{X}}$, we view inverse dialectical flow $\curvearrowleft_\tau$ as a generalization of the concept of consumption, and direct dialectical flow $\curvearrowright_\tau$ as a generalization of the concept of production.

Proposition 1 *Consumption is a special case of inverse dialectical flow. In particular, if the map* $\tau_\theta = (pr_2 \cdot \theta) \oplus d^{\mathrm{sym}} \oplus \tilde{d}$ *associated with any marking* θ *over* X *is a* $\mathbf{V}$*-relation, then ordinary consumption by* θ *is inverse dialectical flow along* τ_θ*; that is,* $\theta \Rightarrow (\,) = \curvearrowleft_{\tau_\theta}$*. Production is a special case of direct dialectical flow. In particular, if the map* $\tau_\theta = (pr_2 \cdot \theta) \oplus \tilde{d}$ *associated with any marking* θ *over* X *is a* $\mathbf{V}$*-relation, then ordinary production by* θ *is direct dialectical flow along* τ_θ*; that is,* $(\,) \oplus \theta = \curvearrowright_{\tau_\theta}$.

For the special case $\mathbf{V} = \mathbf{N}$ when θ is finite everywhere the constraint that τ_θ be a $\mathbf{N}$-relation implies that d is either ∞ or 0 everywhere and that d is ∞ iff e is ∞.

A $\mathbf{V}$-*transition system* A is a triple $\mathsf{A} = \langle T, \mathcal{Q}, \delta\rangle$ consisting of: a set (of transition symbols) T, a quasi $\mathbf{V}$-space (of internal states) $\mathcal{Q}$, and a transition map (transition relation assignment) $\delta : T \longrightarrow \mathbf{rel}_V[\mathcal{Q}, \mathcal{Q}]$. A state pair $(q, q') \in \mathcal{Q}^{\mathrm{op}} \times \mathcal{Q}$ is regarded as an a-transition from current state q to next state q' with weighting (probability, believability, etc.) the generalized truth-value $\delta_a(q, q') \in V$ for each transition symbol $a \in T$. The case $\mathbf{V} = \mathbf{R}$ of (nonnegative) real values includes probabilistic and fuzzy transition systems as special cases. The transition map δ can be recursively extended to a "run map" monoid morphism $\delta^* : T^* \longrightarrow \mathbf{rel}_V[\mathcal{Q}, \mathcal{Q}]$: $\delta^*_\varepsilon = \mathrm{y}_{\mathcal{Q}}$, the identity $\mathbf{V}$-relation at $\mathcal{Q}$; and $\delta^*_{xa} = \delta^*_x \circ \delta_a$ for all strings of transition symbols $x \in T^*$ and single transition symbols $a \in T$. Composing syntactic run map dynamics δ^* with flow $\circledcirc$ defines the transition system dynamics $\delta^* \cdot \circledcirc : T^* \longrightarrow \mathbf{Space}_V[\mathbf{V}^{\mathcal{Q}}, \mathbf{V}^{\mathcal{Q}}]$.

There are three main motivators for the concept of dialectical nets: Petri nets, transition systems and Kan quantification theory of first order predicate logic. The two motivators Petri nets and Kan quatification correspond to the structural concepts of enriched predicates (generalized subobjects) and enriched functions, respectively. The concept of dialectical nets, which generalizes transition systems, corresponds to the structural concept of enriched relations.

An *elementary dialectical* **V**-*net* (or *elementary dialectical* **V**-*transition system*) N is a quadruple $\mathsf{N} = \langle T, \mathcal{S}, \iota, o \rangle$ consisting of: a set (of transition symbols) T, two quasi **V**-spaces (of sites) $\mathcal{S} = \langle \mathcal{S}_0, \mathcal{S}_1 \rangle$, an inverse flow assignment $\iota : T \longrightarrow \mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1]$, and a direct flow assignment $o : T \longrightarrow \mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1]$, where $\mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1]$ is the collection (quasi **V**-space) of all **V**-relations between the quasi **V**-spaces of sites $\mathcal{S}_0$ and $\mathcal{S}_1$. So for each transition symbol $a \in T$ the net assigns a relation pair $\tau_t = \mathcal{S}_0 \overset{o_t}{\underset{\iota_t}{\rightrightarrows}} \mathcal{S}_1$ consisting of a direct flow specifier $\mathcal{S}_0 \xrightarrow{o_t} \mathcal{S}_1$, and an inverse flow specifier $\mathcal{S}_0 \xrightarrow{\iota_t} \mathcal{S}_1$ which specify the dialectical flow $\oslash^{\mathsf{N}}_a = \oslash_{\tau_a} = \circlearrowleft_{\iota_a} \cdot \circlearrowright_{o_a} : \mathbf{V}^{\mathcal{S}_1} \longrightarrow \mathbf{V}^{\mathcal{S}_1}$ for any transition symbol $a \in T$. Dialectical flow $\oslash^{\mathsf{N}}$ can be recursively extended to a "run flow" monoid morphism $\oslash^{\mathsf{N}} : T^* \longrightarrow \mathbf{Space}_V[\mathbf{V}^{\mathcal{S}_1}, \mathbf{V}^{\mathcal{S}_1}]$: $\oslash^{\mathsf{N}}_\varepsilon = \mathrm{Id}_{V^{\mathcal{S}_1}}$, the identity **V**-morphism at $\mathbf{V}^{\mathcal{P}_1}$; and $\oslash^{\mathsf{N}}_{xa} = \oslash^{\mathsf{N}}_x \cdot \oslash^{\mathsf{N}}_a$ for all strings of transition symbols $x \in T^*$ and single transition symbols $a \in T$. When appropriate pullbacks exist, the flow assignment τ can be recursively extended to a "run map" monoid morphism $\tau^* : T^* \longrightarrow \mathbf{rel}_V[\mathcal{S}_1, \mathcal{S}_1]$. A "localized Beck condition" of higher order categorical logic [4] should by incorporated into the above definition of run-flow.

The elements in the quasi **V**-space $\mathcal{S}_1$ are the sites where values (bit-values for conditions in condition/event nets, numbers representing resources in consumption/production nets, database relations in predicate/transition nets, etc.) are stored, and local processing of values (of the nature: combination, accumulation of values, suprema-calculation, union) takes place; whereas, the elements in the quasi **V**-space $\mathcal{S}_0$ are usually sites where transient local processing of values (of the dual nature: interaction, matching of values, infima-calculation, intersection) takes place. In the traditional theory of nets and in the traditional theory of transition systems $\mathcal{S}_0 = \mathcal{S}_1$. In the theory of Horn clause logic programming $\mathcal{S}_1 = \{$predicate (or relational) names in Horn clause logic programs$\}$ and $\mathcal{S}_0 = \{$clauses (or elementary implications) in Horn clause logic programs$\}$. We regard a dialectical net to be a transformer of constrained **V**-markings; that is, a **V**-predicate transformer. The semantics of a dialectical **V**-net N can be defined as either external or internal behaviors. External behaviors include: (1) unfoldment-tree, and (2) regular-set behavior. Internal behaviors include: (1) reachable predicates (markings), and (2) cumulative fixpoint behavior. [There is a formal dialectical approach to net behavior. Define the **V**-relations $v^{\triangleleft} = ((\,) \oplus v)^{\triangleleft} : \mathcal{V} \rightharpoonup \mathcal{V}$ and $v_{\triangleleft} = ((\,) \oplus v)_{\triangleleft} : \mathcal{V} \leftharpoondown \mathcal{V}$. Note that $\Box v_{\triangleleft}(y, x)$ iff $y \preceq x \oplus v$, a generalized net enabling condition. Now $v^{\triangleleft}$ is formally left adjoint to $v_{\triangleleft}$ (as an arrow in the 2-category $\mathbf{rel}_V$), $v^{\triangleleft} \dashv v_{\triangleleft}$. Let $\lhd_V(v)$ denote this adjunction. Then $\lhd_V : \bullet_V \longrightarrow \mathbf{rel}_V$ is a formal dialectical base. Let N be any **V**-net. Define the dialectical flow $\oslash_t = (\iota_t)_{\triangleleft} \circ (o_t)^{\triangleleft}$ for each transition symbol $t \in T$. For normal **V**, $\Box\oslash_t(z, x)$ iff $z \preceq y \oplus \iota_t$ and $y \oplus o_t \preceq x$ some $y \in V$. Define $\oslash_\varepsilon = \mathrm{y}_{\mathcal{V}}$; $\oslash_{xa} = \oslash_x \circ \oslash_a$ for all $x \in T^*$ and $a \in T$; and $\oslash_* = \bigvee_{w \in T^*} \oslash_w$. We say that x is *reachable by* w from y when $\Box\oslash_w(y, x)$; that is, when $e \preceq \oslash_w(y, x)$. We say that x is *reachable* from y when $\Box\oslash_*(y, x)$; that is, when $e \preceq \oslash_*(y, x)$. So $\oslash_*(y, x)$ gives a measure of reachability.] Boundedness, liveness, synchronic distance and fairness are definable for dialectical nets just as for ordinary nets. The dialectical **V**-net N is deterministic when ι and o factor through $\mathrm{Space}_V[\mathcal{S}_0, \mathcal{S}_1]$. Then dialectical flow is existential-quantification/substition composition $\oslash^{\mathsf{N}}_a = \mathbf{V}^{\iota_a} \cdot \exists_{o_a}$. Using the places-as-sites interpretation, any **V**-net $\mathsf{N} = \langle T, \mathcal{P}, \iota, o \rangle$ is an elementary dialectical **V**-net. Indeed, for the special case where $\mathcal{S}_0$ is the terminal-coterminal **V**-space $\mathcal{S}_0 = \mathbf{1}$ and $\mathcal{S}_1$ is the place-space $\mathcal{S}_1 = \mathcal{P}$ (so that $\mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1] = \mathbf{rel}_V[\mathbf{1}, \mathcal{P}] = \mathbf{V}^{\mathcal{P}}$), elementary dialectical **V**-nets $\equiv$ **V**-nets. Any **V**-transition system $\mathsf{A} = \langle T, \mathcal{Q}, \delta \rangle$ is an elementary dialectical **V**-net. Indeed, for the special case where the two site-spaces are the one state-space $\mathcal{S}_0 = \mathcal{Q} = \mathcal{S}_1$, where inverse flow is the trivial identity **V**-relation $\iota_a = \mathrm{y}_{\mathcal{Q}}$ on $\mathcal{Q}$ for all transition symbols $a \in T$, and where direct flow is the transition map $o = \delta$, elementary dialectical **V**-nets $\equiv$ **V**-transition systems. Using the markings-as-fuzzy-subsets interpretation, any **V**-net $\mathsf{N} = \langle T, P, \iota, o \rangle$ is an elementary dialectical $\mathbf{V}^P$-net $\mathsf{N} = \langle T, \mathbf{1}, \iota, o \rangle$ with only one site $\mathcal{S}_0 = \mathbf{1} = \mathcal{S}_1$ (so that $\mathbf{rel}_{V^P}[\mathcal{S}_0, \mathcal{S}_1] = \mathbf{rel}_{V^P}[\mathbf{1}, \mathbf{1}] \cong \mathbf{V}^P$), where inverse flow $\iota : T \longrightarrow \mathbf{V}^P \cong \mathbf{rel}_{V^P}[\mathbf{1}, \mathbf{1}]$ and direct flow $o : T \longrightarrow \mathbf{V}^P \cong \mathbf{rel}_{V^P}[\mathbf{1}, \mathbf{1}]$ assign $\mathbf{V}^P$-elements as $\mathbf{V}^P$-relations. That is, elementary dialectical $\mathbf{V}^P$-nets on one site $\equiv$ **V**-nets. Here $\oslash_t(\mu) = [\iota_t \Rightarrow \mu] \oplus o_t = \mathsf{N}_t(\mu)$ for any transition symbol $t \in T$; that is, dialectical flow is ordinary consumption/production transitioning. These are not transition systems since inverse flow is not identity.

Any two **V**-relations in the collection $\{\mathcal{S}_0 \xrightarrow{\iota_a} \mathcal{S}_1 \mid a \in T\}$ of inverse flow assignments are obviously comparable by use of the inf metric on the quasi **V**-space $\mathbf{V}^{\mathcal{S}_0^{\mathrm{op}} \times \mathcal{S}_1}$. We can intend, or specify, relationships between the ι_a by requiring that T be a general (and not just discrete) quasi **V**-space $\mathcal{T} = \langle T, d_T \rangle$, and that inverse flow assignment be a **V**-morphism $\iota : \mathcal{T}^{\mathrm{op}} \longrightarrow \mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1] = \mathbf{V}^{\mathcal{S}_0^{\mathrm{op}} \times \mathcal{S}_1}$. The same comments apply to the direct flow assignment o. A ***dialectical* V*-net*** (or ***dialectical* V*-transition system***) N is a quadruple $\mathsf{N} = \langle \mathcal{T}, \mathcal{S}, \iota, o \rangle$ consisting of: a quasi **V**-space of transition symbols $\mathcal{T}$, two quasi **V**-spaces of sites $\mathcal{S} = \langle \mathcal{S}_0, \mathcal{S}_1 \rangle$, an inverse flow **V**-morphism $\iota : \mathcal{T}^{\mathrm{op}} \longrightarrow \mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1]$, and a direct flow **V**-morphism $o : \mathcal{T}^{\mathrm{op}} \longrightarrow \mathbf{rel}_V[\mathcal{S}_0, \mathcal{S}_1]$. Dialectical nets have the same run-flow dynamics as elementary dialectical nets. It is clear that every dialectical net $\mathsf{N} = \langle \mathcal{T}, \mathcal{S}, \iota, o \rangle$ defines a (single) dialectical flow specifier. Both the input and the output weighting functions are **V**-morphisms $\iota, o : \mathcal{T}^{\mathrm{op}} \times \mathcal{S}_0^{\mathrm{op}} \times \mathcal{S}_1 \longrightarrow \mathbf{V}$; that is, **V**-relations $\iota, o : \mathcal{T} \times \mathcal{S}_0 \rightharpoondown \mathcal{S}_1$. This means that the dialectical net N is just the (single) relation pair $\mathsf{N} = \mathcal{T} \times \mathcal{S}_0 \underset{\iota}{\overset{o}{\rightrightarrows}} \mathcal{S}_1$, which can be viewed as an enriched *state-space graph* of N. For a **V**-transition system $\mathsf{A} = \langle \mathcal{T}, \mathcal{Q}, \delta \rangle$, $\mathcal{T} \times \mathcal{S}_0 = \mathcal{T} \times \mathcal{Q}$ is the T-th copower of $\mathcal{Q}$. If the transition system A is discrete and deterministic, $o = \delta : T \times Q \longrightarrow Q$ is the usual determistic state transition function, $\iota = \mathrm{pr}_Q : T \times Q \longrightarrow Q$ is the projection function (identity inverse flow), $T \times Q$ is the set of edges in the state-space, and the relation pair $\mathsf{A} = \mathcal{T} \times \mathcal{Q} \underset{\mathrm{pr}_Q}{\overset{\delta}{\rightrightarrows}} \mathcal{Q}$ is the ordinary state-space graph for the transition system A. If we interpret the state-space graph of a dialectical net N to be a dialectical flow specifier we can define an aggregate dialectical flow $\circlearrowleft^{\mathsf{N}} : \mathbf{V}^{\mathcal{S}_1} \longrightarrow \mathbf{V}^{\mathcal{S}_1}$ on **V**-predicates over the site-space $\mathcal{S}_1$. This aggregate dialectical flow, and its various fixpoints, define a combined external-internal behavior for the dialectical net N. For a fixed space of transition symbols $\mathcal{T}$, given any two dialectical **V**-nets $\langle \mathcal{S}_0, \tau_0 \rangle = \langle \mathcal{S}_0, \iota_0, o_0 \rangle$ and $\langle \mathcal{S}_1, \tau_1 \rangle = \langle \mathcal{S}_1, \iota_1, o_1 \rangle$, a *morphism* of dialectical **V**-nets $\langle h_0, h_1 \rangle : \langle \mathcal{S}_0, \iota_0, o_0 \rangle \Rightarrow \langle \mathcal{S}_1, \iota_1, o_1 \rangle$ consists of two **V**-morphisms $h_0 : \mathcal{S}_{0,0} \longrightarrow \mathcal{S}_{1,0}$ and $h_1 : \mathcal{S}_{0,1} \longrightarrow \mathcal{S}_{1,1}$, where $\langle h_0, h_1 \rangle : \langle \mathcal{S}_0, \tau_{0,a} \rangle \Rightarrow \langle \mathcal{S}_1, \tau_{1,a} \rangle$ is a vertical morphism in $\mathbf{Rel}_V^{\times}$ for all transition symbols $a \in T$. For fixed $\mathcal{T}$, dialectical **V**-nets and their morphisms form a category $\mathbf{Net}_V^{\mathcal{T}}$, the $\mathcal{T}$-th fiber of the *category of dialectical* **V***-nets* $\mathbf{Net}_V$.

DIALECTICAL SYSTEMS

In this section we show that each dialectical net determines a special kind of dialectical system. Let $\mathbf{V} = \langle V, \preceq, \oplus, \Rightarrow, e \rangle$ be any closed preorder. The monoid $\langle V, \oplus, e \rangle$ can be regarded to be a category, denoted by $\bullet_V$. Let us give the monotonic functions of **V**-composition and **V**-implication the more function-like notation $V^v(\,) = (\,) \oplus v : V \longrightarrow V$ and $V_v(\,) = v \Rightarrow (\,) : V \longrightarrow V$ for each **V**-element $v \in V$. Then the closure axiom becomes the adjunction statement $(V^v \dashv V_v) : V \longrightarrow V$ for each **V**-element $v \in V$. Let $\mathbf{V}(n)$ denote this adjunction. In objective dialectics, since dialectical contradictions are represented by adjunctions, systems of dialectical contradictions are represented by diagrams (pseudofunctors) in the category **adj** whose objects are bicomplete preorders and whose morphisms are adjoint pairs of monotonic functions. We call such a (pseudo)functor $\mathbf{P} : \mathbf{\Omega} \longrightarrow \mathbf{adj}$ a *dialectical base* of preorders, and use the notation $\mathbf{P}(w_1 \xrightarrow{t} w_2) = (\mathbf{P}^t \dashv \mathbf{P}_t) : \mathbf{P}_{w_1} \rightarrow \mathbf{P}_{w_2}$. Objects of $\mathbf{\Omega}$ are called *types* and arrows of $\mathbf{\Omega}$ are called *terms*. Dialectical systems are the "motors of nature" specifying the dialectical motion of structured entities, and a dialectical base provides the "motive power" for this motion. The entire set of axioms for the closed preorder $\mathbf{V} = \langle V, \preceq, \oplus, \Rightarrow, e \rangle$ are equivalent to the following single statement.

Fact 2 *The operator* $\mathbf{V}(\,)$ *is a dialectical base* $\mathbf{V} : \bullet_V \longrightarrow \mathbf{adj}$.

We now develop an equivalent fibrational approach for formalizing the dialectical structure of the **V**-elements. Any quasi **V**-space $\mathcal{X} = \langle X, d_X \rangle$ determines a category $\mathbf{X}$, whose objectset is $\mathrm{Obj}(\mathbf{X}) = X$, whose arrowset is $\mathrm{Ar}(\mathbf{X}) = \{(x, v, x') \mid v \preceq d_X(x, x')\}$ with homsets being the principal ideals $\mathbf{X}[x, x'] \cong {\downarrow_V}\, d_X(x, x')$, whose source and target functions are the projections $\mathrm{pr}_1, \mathrm{pr}_3 : \mathrm{Ar}(\mathbf{X}) \longrightarrow X$, whose identities are $\mathrm{Id}_x = (x, e, x)$ for each $x \in X$, and whose composition is $(x, v, x') \circ (x', v', x'') = (x, v \oplus v', x'')$. Clearly the projection function $\mathrm{pr}_2 : \mathrm{Ar}(\mathbf{X}) \longrightarrow V$ defines a functor $|\,|_{\mathcal{X}} = \mathrm{pr}_2 : \mathbf{X} \longrightarrow \bullet_V$. Since (x, e, x') is an $\mathbf{X}$-arrow iff $e \preceq d_X(x, x')$ iff $x \preceq_X x'$, the fiber of $|\,|_{\mathcal{X}}$ over the (only) $\bullet_V$-object e is $|\,|_{\mathcal{X},e} = |\,|_{\mathcal{X}}^{-1}(e) = \{(x, e, x') \mid e \preceq d_X(x, x')\} \cong \{(x, x') \mid x \preceq_X x'\}$; that is, the fiber $|\,|_{\mathcal{X},e}$ is essentially the preorder $\langle X, \preceq_X \rangle$ viewed as a subcategory of $\mathbf{X}$. A *distributed* **V***-type* (or a **V***-normed category*) $\mathcal{C}$ is a pair $\mathcal{C} = \langle \mathbf{C}, |\,|_{\mathcal{C}} \rangle$ where $\mathbf{C}$ is a category and $|\,|_{\mathcal{C}} : \mathbf{C} \longrightarrow \bullet_V$ is a functor called a **V***-norm* (or a **V***-typing*). So every quasi **V**-space $\mathcal{X} = \langle X, d_X \rangle$ determines a distributed **V**-type $\mathcal{X} = \langle \mathbf{X}, |\,|_{\mathcal{X}} \rangle$. The category **V** of **V***-inequality conditions* is the category part of $\langle \mathbf{V}, |\,|_{\mathcal{V}} \rangle$, the distributed **V**-type determined by the quasi **V**-space $\mathcal{V}$ of generalized truth-values. The functor part $|\,|_{\mathcal{V}}$ has special properties, and so we give it a special notation: $P_V = |\,|_{\mathcal{V}}$. By the above fact, P_V is a 01-fibration. The only fiber of P_V is $P_{V,e} \cong \mathcal{V} = \langle V, \preceq \rangle$ which is bicomplete (a cpo).

Proposition 2 **V***-inequality conditions form a dialectical base over* **V***-truth-values; that is, the* **V***-norm* $P_V : \mathbf{V} \longrightarrow \bullet_V$ *is a dialectical base.*

We can extend this result from $\mathbf{V} = \langle \mathcal{V}, \oplus, \Rightarrow \rangle$ itself to any tensored-cotensored quasi **V**-space . A **V**-predicate $\phi \in \mathbf{V}^{\mathcal{X}}$ over a quasi **V**-space $\mathcal{X}$ is a **V**-morphism $\phi : \mathcal{X} \longrightarrow \mathbf{V}$, and hence determines a morphism of distributed **V**-types $F_{\phi} : \mathrm{Type}_{\mathcal{V}}(\mathcal{X}) \longrightarrow \mathrm{Type}_{\mathcal{V}}(\mathcal{V})$; that is, a functor $F_{\phi} : \mathbf{X} \longrightarrow \mathbf{V}$ satisfying $F_{\phi} \cdot P_{\mathcal{V}} = |\,|_{\mathcal{X}}$. Given a distributed **V**-type $\mathcal{C} = \langle \mathbf{C}, |\,|_{\mathcal{C}} \rangle$, a *distributed* **V**-*entity* (or **V**-*predicate*) Φ of type $\mathcal{C}$ is a functor $F : \mathbf{C} \longrightarrow \mathbf{V}$ satisfying $F \cdot P_{\mathcal{V}} = |\,|_{\mathcal{C}}$. Let $\mathbf{V}^{\mathcal{C}}$ denote the collection (bicomplete quasi **V**-space) of all distributed **V**-entities of type $\mathcal{C}$.

Given two categories **B** and **A**, a *distributer* **R** from category **A** to category **B**, denoted by $\mathbf{B} \xrightarrow{\mathbf{R}} \mathbf{A}$, is a triple $\mathbf{R} = \langle \circ_0, R, \circ_1 \rangle$, where: R is a span $R = \langle |\mathbf{B}|, \partial_0, \mathrm{Ar}(\mathbf{R}), \partial_1, |\mathbf{A}| \rangle$ with arrowset $\mathrm{Ar}(\mathbf{R})$, source function $\partial_0 : \mathrm{Ar}(\mathbf{R}) \longrightarrow |\mathbf{B}|$, and target function $\partial_1 : \mathrm{Ar}(\mathbf{R}) \longrightarrow |\mathbf{A}|$; $\circ_0$ is a left action with respect to **B**, so that $\mathrm{Id}_b \circ_0 e = e$ and $(g' \circ_B g) \circ_0 e = g' \circ_0 (g \circ_0 e)$ for all R-arrows $b \xrightarrow{e} a$ and all **B**-arrows $b'' \xrightarrow{g'} b'$ and $b' \xrightarrow{g} b$; $\circ_1$ is a right action with respect to **A**, so that $e \circ_1 \mathrm{Id}_a = e$ and $e \circ_1 (f \circ_A f') = (e \circ_1 f) \circ_1 f'$ for all R-arrows $b \xrightarrow{e} a$ and all **A**-arrows $a \xrightarrow{f} a'$ and $a' \xrightarrow{f'} a''$; and the mixed associative law $g \circ_0 (e \circ_1 f) = (g \circ_0 e) \circ_1 f)$ holds. A category **C** is a distributer $\mathbf{C} \xrightarrow{\mathrm{hom}_C} \mathbf{C}$ on **C**. Categories and distributers form a category **Dist**, which includes as a subcategory (via Yoneda) the category **Cat** of categories and functors. A *morphism of distributers* $\mathbf{R}_1 \xrightarrow{F} \mathbf{R}_2$ from distributer $\mathbf{B}_1 \xrightarrow{\mathbf{R}_1} \mathbf{A}_1$ to distributer $\mathbf{B}_2 \xrightarrow{\mathbf{R}_2} \mathbf{A}_2$, is a triple $F = \langle g, h, f \rangle$ consisting of a pair of functions between objectsets $g : |\mathbf{B}_1| \longrightarrow |\mathbf{B}_2|$, $f : |\mathbf{A}_1| \longrightarrow |\mathbf{A}_2|$ and a function between arrowsets $h : \mathrm{Ar}(\mathbf{R}_1) \longrightarrow \mathrm{Ar}(\mathbf{R}_2)$ which satisfy the compatability conditions $\partial_0^1 \cdot g = h \cdot \partial_0^2$ and $\partial_1^1 \cdot f = h \cdot \partial_1^2$. A functor $\mathbf{C} \xrightarrow{F} \mathbf{D}$ is a morphism of distributers, with categories **C** and **D** regarded as distributers, where $F = \langle |F| : |\mathbf{C}| \rightarrow |\mathbf{D}|, F : \mathrm{Ar}(\mathbf{C}) \rightarrow \mathrm{Ar}(\mathbf{D}), |F| : |\mathbf{C}| \rightarrow |\mathbf{D}| \rangle$.

Any **V**-relation $\mathcal{Y} \xrightarrow{\tau} \mathcal{X}$ determines a distributer $\mathbf{Y} \xrightarrow{\mathbf{T}} \mathbf{X}$ whose arrowset is $\mathrm{Ar}(\mathbf{T}) = \{(y, v, x) \mid y \in Y, x \in X, v \preceq \alpha(y, x)\}$ with homsets being the principal ideals $\mathbf{T}[y, x] \cong \downarrow_{\mathcal{V}} \alpha(y, x)$, whose source and target functions are the projections $\partial_0^T = \mathrm{pr}_1 : \mathrm{Ar}(\mathbf{T}) \longrightarrow |\mathbf{Y}|$ and $\partial_1^T = \mathrm{pr}_3 : \mathrm{Ar}(\mathbf{T}) \longrightarrow |\mathbf{X}|$, and whose biaction consists of the left action $(y', v, y) \circ_0 (y, w, x) = (y', v \oplus w, x)$ and the right action $(y, w, x) \circ_1 (x, u, x') = (y, w \oplus u, x')$ (essentially the **V**-composition $\oplus$ distributed over Y and X). Clearly the projection function $\mathrm{pr}_2 : \mathrm{Ar}(\mathbf{T}) \longrightarrow V$ defines a morphism of distributers $|\,|_{\mathcal{T}} = \mathrm{pr}_2 : \mathbf{T} \longrightarrow \bullet_{\mathcal{V}}$. Since (y, e, x) is an **T**-arrow iff $e \preceq \alpha(y, x)$ iff $y \preceq_{\alpha} x$, the fiber of $|\,|_{\mathcal{T}}$ over the (only) $\bullet_{\mathcal{V}}$-object e is $|\,|_{\mathcal{T},e} = |\,|_{\mathcal{T}}^{-1}(e) = \{(y, e, x) \mid e \preceq \alpha(y, x)\} \cong \{(y, x) \mid y \preceq_{\alpha} x\}$; that is, the fiber $|\,|_{\mathcal{X},e}$ is essentially the **2**-relation $\mathcal{Y} \xrightarrow{\preceq_{\alpha}} \mathcal{X}$ viewed as a subdistributer of $\mathbf{T} = (\mathbf{Y} \xrightarrow{T} \mathbf{X})$. A *distributed* **V**-*term* (or a **V**-*normed distributer*) $\mathcal{R}$ is a pair $\mathcal{R} = \langle \mathbf{R}, |\,|_{\mathcal{R}} \rangle$ where $\mathbf{B} \xrightarrow{\mathbf{R}} \mathbf{A}$ is a distributer and $|\,|_{\mathcal{R}} : \mathbf{R} \longrightarrow \bullet_{\mathcal{V}}$ is a morphism of distributers called a **V**-*norm* (or a **V**-*typing*). So every **V**-relation $\mathcal{Y} \xrightarrow{\tau} \mathcal{X}$ determines a distributed **V**-term $\mathcal{T} = \langle \mathbf{T}, |\,|_{\mathcal{T}} \rangle$. Let $\mathrm{Term}_{\mathcal{V}}(\tau) = \langle \mathbf{T}, |\,|_{\mathcal{T}} \rangle$ denote this construction. Any morphism $\tau_1 \xrightarrow{\langle g, f \rangle} \tau_2$ of **V**-relations $\mathcal{Y}_1 \xrightarrow{\tau_1} \mathcal{X}_1$ and $\mathcal{Y}_2 \xrightarrow{\tau_2} \mathcal{X}_2$ determines a morphism of distributers $\mathcal{T}_1 \xrightarrow{F_{g,f}} \mathcal{T}_2$ where $F_{g,f} = \langle g, h, f \rangle$ and $h((y, v, x)) = (g(y), v, f(x))$ on $\mathbf{T}_1$-arrows. Clearly $F_{g,f}$ commutes with the projection functions: $F_{g,f} \cdot |\,|_{\mathcal{T}_2} = |\,|_{\mathcal{T}_1}$. Let $\mathrm{Term}_{\mathcal{V}}(\mathbf{A}) = \mathrm{Space}(\mathbf{A})$ and $\mathrm{Term}_{\mathcal{V}}(\langle g, f \rangle) = F_{g,f}$ denote this construction. A *morphism* of distributed **V**-terms $F : \langle \mathbf{R}, |\,|_{\mathcal{R}} \rangle \longrightarrow \langle \mathbf{S}, |\,|_{\mathcal{S}} \rangle$ is a morphism of distributers $F : \mathbf{R} \longrightarrow \mathbf{S}$ which commutes with the **V**-norms: $F \cdot |\,|_{\mathcal{R}} = |\,|_{\mathcal{S}}$. So every **V**-relation morphism is a distributed **V**-term morphism. Let $\mathbf{Term}_{\mathcal{V}}$ denote the category of distributed **V**-terms and their morphisms. $\mathbf{Term}_{\mathcal{V}}$ is the comma category $\mathbf{Term}_{\mathcal{V}} = \mathbf{Dist} \downarrow \bullet_{\mathcal{V}}$. Then $\mathrm{Term}_{\mathcal{V}}$ is a functor $\mathrm{Term}_{\mathcal{V}} : \mathbf{Rel}_{\mathcal{V}} \longrightarrow \mathbf{Term}_{\mathcal{V}}$ from relations to terms.

In the opposite direction, any distributed **V**-term $\mathcal{T} = \langle (\mathbf{B} \xrightarrow{\mathbf{T}} \mathbf{A}), |\,|_{\mathcal{T}} \rangle$ determines a **V**-relation $\mathrm{Space}(\mathbf{B}) \xrightarrow{\tau_{\mathcal{T}}} \mathrm{Space}(\mathbf{A})$, where the **V**-morphism $\tau_{\mathcal{T}} : B \times A \longrightarrow V$ is defined to be the homset supremum $\tau_{\mathcal{T}}(b, a) = \bigvee_{b \xrightarrow{e} a} |e|_{\mathcal{T}}$. Let $\mathrm{Rel}_{\mathcal{V}}(\langle \mathbf{T}, |\,|_{\mathcal{T}} \rangle) = \tau_{\mathcal{T}}$ denote this construction. Any morphism of distributed **V**-terms $F = \langle g, h, f \rangle : \mathcal{T}_1 \longrightarrow \mathcal{T}_2$, where $\mathcal{T}_1 = \langle \mathbf{T}_1, |\,|_{\mathcal{T}_1} \rangle$ and $\mathcal{T}_2 = \langle \mathbf{T}_2, |\,|_{\mathcal{T}_2} \rangle$, determines a morphism of **V**-relations $\langle g, f \rangle : \mathrm{Rel}_{\mathcal{V}}(\mathcal{T}_1) \longrightarrow \mathrm{Rel}_{\mathcal{V}}(\mathcal{T}_2)$. Let $\mathrm{Rel}_{\mathcal{V}}(F) = \langle g, f \rangle$ denote this construction. Then $\mathrm{Rel}_{\mathcal{V}}$ is a functor $\mathrm{Rel}_{\mathcal{V}} : \mathbf{Term}_{\mathcal{V}} \longrightarrow \mathbf{Rel}_{\mathcal{V}}$ from terms to relations. For any distributed **V**-term $\mathcal{T} = \langle \mathbf{T}, |\,|_{\mathcal{T}} \rangle$ there is a canonical morphism of **V**-terms $\eta_{\mathcal{T}} : \mathcal{T} \longrightarrow \mathrm{Term}_{\mathcal{V}}(\mathrm{Rel}_{\mathcal{V}}(\mathcal{T}))$, where $\eta_{\mathcal{T}} = \langle \mathrm{Id}_B, h, \mathrm{Id}_A \rangle$ and $h(b \xrightarrow{e} a) = b \xrightarrow{(b, |e|_{\mathcal{T}}, a)} a$ on arrows. Moreover, $\mathrm{Rel}_{\mathcal{V}}(\mathrm{Term}_{\mathcal{V}}(\tau)) = \tau$ for any **V**-relation τ.

Proposition 3 *The* **V**-*relation functor is left adjoint* $\mathrm{Rel}_{\mathcal{V}} \dashv \mathrm{Term}_{\mathcal{V}}$ *to the* **V**-*term functor.*

The unit of this adjunction $\eta : \mathrm{Id} \Longrightarrow \mathrm{Rel}_{\mathcal{V}} \cdot \mathrm{Term}_{\mathcal{V}}$ has the canonical morphism of **V**-terms $\eta_{\mathcal{T}}$ as its $\mathcal{T}$-th component, and the counit of this adjunction is the identity natural transformation $\mathrm{Id} : \mathrm{Term}_{\mathcal{V}} \cdot \mathrm{Rel}_{\mathcal{V}} \Longrightarrow \mathrm{Id}$. So this adjunction is a reflection with $\mathrm{Term}_{\mathcal{V}} : \mathbf{Rel}_{\mathcal{V}} \longrightarrow \mathbf{Term}_{\mathcal{V}}$ embedding $\mathbf{Rel}_{\mathcal{V}}$ as a subcategory of $\mathbf{Term}_{\mathcal{V}}$, and $\mathrm{Rel}_{\mathcal{V}} : \mathbf{Term}_{\mathcal{V}} \longrightarrow \mathbf{Rel}_{\mathcal{V}}$ reflecting $\mathbf{Term}_{\mathcal{V}}$ into its "subcategory" $\mathbf{Rel}_{\mathcal{V}}$.

Let $\mathbf{P} : \boldsymbol{\Omega} \longrightarrow \mathbf{adj}$, or equivalently, $P_P : \mathbf{P} \longrightarrow \boldsymbol{\Omega}$ be any dialectical base. A *distributed* **P**-*type* $\mathcal{C}$

is a pair $\mathcal{C} = \langle \mathbf{C}, |\,|_{\mathcal{C}} \rangle$ where $\mathbf{C}$ is a category and $|\,|_{\mathcal{C}} : \mathbf{C} \longrightarrow \mathbf{\Omega}$ is a functor called a **P**-*typing*. Given a distributed **P**-type $\mathcal{C} = \langle \mathbf{C}, |\,|_{\mathcal{C}} \rangle$, a *distributed* **P**-*entity* Φ of type $\mathcal{C}$ is a functor $F : \mathbf{C} \longrightarrow \mathbf{P}$ satisfying $F \cdot P_P = |\,|_{\mathcal{C}}$. Let $\mathbf{P}^{\mathcal{C}}$ denote the collection (bicomplete preorder) of all distributed **P**-entities of type $\mathcal{C}$. A *distributed* **P**-*term* (or a **P**-*typed distributer*) $\mathcal{R}$ is a pair $\mathcal{R} = \langle \mathbf{R}, |\,|_{\mathcal{R}} \rangle$ where $\mathbf{B} \overset{\mathbf{R}}{\rightarrow} \mathbf{A}$ is a distributer and $|\,|_{\mathcal{R}} : \mathbf{R} \longrightarrow \mathbf{\Omega}$ is a morphism of distributers called a **P**-*typing*. Let $\mathbf{Term}_P$ denote the category of distributed **P**-terms and their morphisms. Any distributed **P**-term $\mathcal{R} = \langle \mathbf{R}, |\,|_{\mathcal{R}} \rangle$ with distributer $\mathbf{B} \overset{\mathbf{R}}{\rightarrow} \mathbf{A}$ and **P**-*typing* $|\,|_{\mathcal{R}} : \mathbf{R} \longrightarrow \mathbf{\Omega}$ defines by composition the **P**-morphism $\mathbf{P}^{\mathcal{B}} \xrightarrow{\curvearrowright_{\mathcal{R}}} \mathbf{P}^{\mathcal{A}}$ called *direct flow* (or *yang*), where $\curvearrowright_{\mathcal{R}}$ is defined by $\curvearrowright_{\mathcal{R}}(\Psi)(A) = \bigvee_{B \in |\mathbf{B}|, e \in \mathbf{R}[B,A]} \mathbf{P}^{|e|_{\mathcal{R}}}(\Psi(B))$ for any distributed **P**-entity $\Psi \in \mathbf{P}^{\mathcal{B}}$ of distributed **P**-type $\mathcal{B} = \langle \mathbf{B}, |\,|_{\mathcal{B}} \rangle$ and any object $A \in |\mathbf{A}|$, where $\bigvee$ denotes supremum, or colimit, in $\mathbf{P}(|A|_{\mathcal{A}})$. The **P**-morphism $\curvearrowright_{\mathcal{R}}$ has as a right adjoint the **P**-morphism $\mathbf{P}^{\mathcal{B}} \xleftarrow{\curvearrowleft_{\mathcal{R}}} \mathbf{P}^{\mathcal{A}}$ called *inverse flow* (or *yin*), and defined by $\curvearrowleft_{\mathcal{R}}(\Phi)(B) = \bigwedge_{A \in |\mathbf{A}|, e \in \mathbf{R}[B,A]} \mathbf{P}_{|e|_{\mathcal{R}}}(\Phi(A))$ for any distributed **P**-entity $\Phi \in \mathbf{P}^{\mathcal{A}}$ of distributed **P**-type $\mathcal{A} = \langle \mathbf{A}, |\,|_{\mathcal{A}} \rangle$ and any object $B \in |\mathbf{B}|$, where $\bigwedge$ denotes infimum, which is limit, in $\mathbf{P}(|B|_{\mathcal{B}})$. Parallel pairs of distributed **P**-terms specify "the dialectical motion (flow, development) of entities".

Fact 3 *Direct flow is left adjoint to inverse flow* $\curvearrowright_{\mathcal{R}} \dashv \curvearrowleft_{\mathcal{R}}$ *for any distributed* **P**-*term* $\mathcal{R}$.

A parallel pair $\tau = \mathcal{B} \underset{\mathcal{I}}{\overset{\mathcal{O}}{\rightrightarrows}} \mathcal{A}$ of distributed **P**-terms is called a *dialectical flow specifier*. Given a flow specifier τ, the *dialectical flow* (or *yinyang*) specified by τ is the composition $\circlearrowleft_{\tau} = \mathbf{P}^{\mathcal{A}} \xrightarrow{\curvearrowleft_{\mathcal{I}}} \mathbf{P}^{\mathcal{B}} \xrightarrow{\curvearrowright_{\mathcal{O}}} \mathbf{P}^{\mathcal{A}}$. We say that τ *reproduces* the distributed **P**-entity $\Phi \in \mathbf{P}^{\mathcal{A}}$ when Φ is a fixpoint solution of the recursive equation $\chi \equiv \circlearrowleft_{\tau}(\chi)$. An *elementary dialectical* **P**-*system* S is a quadruple $\mathsf{S} = \langle T, \mathcal{S}, \iota, o \rangle$ consisting of: a set (of transition symbols) T, two distributed **P**-types (of sites) $\mathcal{S} = \langle \mathcal{S}_0, \mathcal{S}_1 \rangle$, an inverse flow assignment $\mathcal{I} : T \longrightarrow \mathbf{Term}_P[\mathcal{S}_0, \mathcal{S}_1]$, and a direct flow assignment $O : T \longrightarrow \mathbf{Term}_P[\mathcal{S}_0, \mathcal{S}_1]$, where $\mathbf{Term}_P[\mathcal{S}_0, \mathcal{S}_1]$ is the collection of all distributed **P**-terms between the distributed **P**-types of sites $\mathcal{S}_0$ and $\mathcal{S}_1$. For fixed category of transition symbols $\mathbf{T}$, we can define morphisms of dialectical **P**-systems, analogous to those for nets. Then, for fixed $\mathbf{T}$, dialectical **P**-systems and their morphisms form a category $\mathbf{kosmos}_{\mathbf{P}}^{\mathbf{T}}$, the $\mathbf{T}$-th fiber of the *category of dialectical* **P**-*systems* $\mathbf{kosmos}_{\mathbf{P}}$ (the **P**-th kosmos).

If $\mathbf{V}$ is any closed preorder with associated dialectical base $\mathbf{V} : \bullet_V \longrightarrow \mathbf{adj}$, or equivalently $P_V : \mathbf{V} \longrightarrow \bullet_V$, then dialectical **V**-systems (as shown above) are (or more precisely, can be reflected into) dialectical **V**-nets. If $D \cdot \mathbf{P}_{\mathbf{Set}} : \mathbf{T}_{\Sigma}^{\mathrm{op}} \rightarrow \mathbf{Set} \rightarrow \mathbf{adj}$ is the dialectical base, where Σ is an algebraic signature with term category $\mathbf{T}_{\Sigma}$, D is a Σ-algebra in functorial form, and $\mathbf{P}_{\mathbf{Set}} : \mathbf{Set} \longrightarrow \mathbf{adj}$ is the direct-image/inverse-image pseudofunctor, then dialectical $D \cdot \mathbf{P}_{\mathbf{Set}}$-systems are Horn clause logic programs. Horn clause logic programs can be enriched by replacing the pseudofunctor $\mathbf{P}_{\mathbf{Set}} : \mathbf{Set} \longrightarrow \mathbf{adj}$ with the existential-Kan-quantification/substitution pseudofunctor $\mathbf{P}_{\mathbf{V}} : \mathbf{Space}_V \longrightarrow \mathbf{adj}$, where $\mathbf{P}_{\mathbf{V}}(\mathcal{X} \xrightarrow{f} \mathcal{Y}) = \mathbf{V}^{\mathcal{X}} \underset{\mathbf{V}^{f}}{\overset{\exists_f}{\rightleftarrows}} \mathbf{V}^{\mathcal{Y}}$, and defining the dialectical base to be $D \cdot \mathbf{P}_{\mathbf{V}} : \mathbf{T}_{\Sigma}^{\mathrm{op}} \rightarrow \mathbf{Space}_V \rightarrow \mathbf{adj}$, where D is any **V**-enriched Σ-algebra (see [3] for further development of this case). In particular, the special case of natural numbers $\mathbf{V} = \mathbf{N}$ enriches Horn clause logic programs with multiplicities, and gives a proper formulation for "predicate/transition nets", which are not just nets, but full-fledged dialectical $D \cdot \mathbf{P}_{\mathbf{N}}$-systems.

REFERENCES

[1] S. Bernow and P. Raskin, Ecology of Scientific Consciousness. Telos 28, Summer (1976).

[2] R.E. Kent, The Metric Closure Powerspace Construction. Third Workshop on the Mathematical Foundations of Programming Language Semantics (1987), Tulane University. LNCS (1988), Springer-Verlag, New York.

[3] R.E. Kent, Dialectical Development in First Order Logic: II. Horn Clause Logic Programming. Manuscript to be submitted to the Logic in Computer Science conference 1988.

[4] F.W. Lawvere, Equality in Hyperdoctrines and the Comprehension Schema as an Adjoint Functor. In, Proceedings New York Symposium on Applications of Categorical Logic (A. Heller, ed.), American Mathematical Society (1970), 1-14.

[5] F.W. Lawvere, Metric Spaces, Generalized Logic, and Closed Categories. Seminario Matematico E. Fisico. Rendiconti. Milan. 43 (1973), 135-166.

On Exact, Critical and Unique Versions of Some Graph Problems*
(Extended Abstract)

Louis E. Rosier
Dept. of Computer Science
University of Texas at Austin
Austin, TX 78712

Hsu-Chun Yen
Dept. of Computer Science
Iowa State University
Ames, IA 50011

Abstract

In this paper, we investigate the complexity results concerning the exact, critical and unique solutions of some graph problems. This study also reveals how the complexity results vary when different constraints are imposed on a problem.

1 Introduction

A fundamental tool used for categorizing the complexity of decision problems is the notion of *completeness.* Over the past decade, abundant numbers of problems have been shown to be complete for NL, P, NP and PSPACE[1]. See e.g. Cook [3], Garey and Johnson [4], and Karp [10]. Recently, some "natural" problems have been shown to be complete for certain classes which are "between" NP and PSPACE (Bentley, Ottmann and Widmayer [1], Papadimitriou [14] and Papadimitriou and Yannakakis [15]) and "between" NL and P (Howell, Rosier and Yen [6] . Lange [11], and Rosier and Yen [17,18]). In [15], Papadimitriou and Yannakakis introduced the class D^P, the class of languages that are intersections of a language in NP and a language in Co-NP. (More precvisely, $D^P = \{L_1 \cap L_2 \mid L_1 \in NP \text{ and } L_2 \in Co-NP\}$.) It is not hard to see NP, Co-NP $\subseteq D^P \subseteq$ PSPACE. D^P is, somewhat, interesting because for many problems that are known to be NP-complete, their "exact" or "critical" versions are complete for this class. An example is the *exact clique* problem. It is known that the existence version of the clique problem, i.e., given a graph, determining whether the graph contains a clique, is complete for NP. However, the exact clique problem (the problem of, given a graph and an integer k, determining whether the maximum clique in the graph has exactly k nodes) is complete for D^P. In view of this, D^P seems to be an interesting complexity class on its own right because it allows us to characterize problems concerning "exact" solutions in a precise manner. (See [15] for other problems complete for D^P.) The question of whether an analogous class exists in sublinear space, therefore, arises. In this paper, we examine the following versions of some graph problems:

---

*Recently, Immerman has shown that NL=Co-NL – as a result, some of the results shown here are now trivial to show. See the concluding remarks.

[1] For brevity, NL and P stand for the complexity classes NLOGSPACE and PTIME, respectively.

1. Existence problems,
2. Exact problems,
3. Critical problems, and
4. Unique solution problems.

It turns out that some problems are complete for the logspace analogy to D^P, namely, D^L – the class of languages that are intersections of a language in NL and a language in Co-NL. (Note that $D^L = \{L_1 \cap L_2 \mid L_1 \in NL \text{ and } L_2 \in Co-NL\}$.) This, perhaps, indicates that D^L is also a "robust" class in the sense that it contains several natural complete problems. The significance of this study is that we are able to pigeonhole several "natural" problems into their proper complexity classes. Furthermore, it also reveals how the complexity results vary when different constraints are imposed on a problem.

The remainder of this paper is organized as follows. In Section 2, we define the problems and illustrate the relationships between the various complexity classes discussed in this paper. In Section 3, complexity results for these problems are derived.

2 Definitions and Problems

Before proceding, some preliminaries concerning graphs are needed. Let G=(V,E) be a directed graph, where V and E are the sets of vertices and edges, respectively. (In this paper, all graphs are directed graphs unless otherwise specified.) Given two vertices u and v, $u \leadsto v$ (v is reachable from u) iff there exists a sequence of vertices $n_0, n_1, ..., n_k$, for some k, such that $u = n_0$, $v = n_k$ and $(n_i, n_{i+1}) \in E$, for $0 \leq i < k$. $u \not\leadsto v$ if no such path exists. A *knot* G′=(V′,E′) is a strongly connected subgraph of G such that for every vertex v ∈ V, if v is reachable from some vertex in V′, then v ∈ V′. A *maximum knot* is a knot with the maximal number of vertices. Knots can be generalized to labelled graphs. Given a labelled graph G=(V,E) where edge labels are over a finite set Σ, a *labelled knot* G′=(V′,E′) is a strongly connected labelled subgraph of G such that for every vertex v ∈ V′, if (v,u), for some u, with label l ($\in \Sigma$) is in E′, then every edge (v,u′) in E whose label is l must be in E′. Given a directed graph G and two vertices x and y, a vertex c is a *cut point* with respect to x and y iff $x \leadsto y$, and $x \not\leadsto y$ when c is removed from G.

Knots, labelled knots and cut points are interesting concepts for many real-world problems. To see this, first observe that a conventional way to describe a distributed communication system is to represent the system by a directed graph, where each vertex indicates a communication entity and each directed edge represents a communication channel between neighboring entities. Suppose A and B are two communication entities. A can forward messages to B providing there is a direct path from A to B. However, if there exists a cut point (with respect to A and B) in the network, then the failure of the cut point will result in a disconnection of the communication between A and B. Thus, to some degree the concept of cut points directly reflects the reliability of the communication in a network. The concept of knot has been used in [13] to deal with the deadlock detection problem for distributed systems. In a concurrent system, it was shown in [18] that labelled knots were a useful concept for analyzing whether the system will terminate under a fair schedule.

In this paper, we consider the following problems:

1. Existence Problems:
    - E-CUTP: Given a graph and two vertices x and y, does there exist a cut point?
    - E-KNOT: Given a graph, does there exist a knot?
    - E-LKNOT: Given a labelled graph, does there exist a labelled knot?
2. Exact Problems:
    - X-KNOT: Given a graph G and an integer k, does the maximum knot contain exactly k vertices?
    - X-LKNOT: Given a graph G and an integer k, does the maximum labelled knot contain exactly k vertices?
3. Critical Problems:
    - C-CUTP: Given a graph G and two vertices x and y, is it true that G does not have a cut point with respect to x and y, but deleting a vertex from it creates a cut point?
    - C-KNOT: Given a graph G, is it true that G does not have knots, but deleting an edge from it creates a knot?
    - C-LKNOT: Given a labelled graph G, is it true that G does not have labelled knots, but deleting an edge from it creates a labelled knot?
4. Unique Solution Problems:
    - U-CUTP: Given a graph and two vertices x and y, is there a unique cut point of x and y?
    - U-KNOT: Given a graph G, does G contain a unique knot?
    - U-LKNOT: Given a labelled graph G, does G contain a unique labelled knot?

In what follows, we assume the reader is familiar with the basic tenets of automata and complexity theory, and complexity classes such as DL, NL, P, etc. Relevant sources would include [4,5]. The basic computational model used in this paper is the *nondeterministic off-line multitape Turing machine* (NTM). All completeness classes mentioned in this paper are with respect to deterministic logspace many-one reductions. We also require the following variation of NTMs and the corresponding complexity classes. In [2], the *alternating logspace hierarchy* was introduced based on the model of *alternating Turing machines* (ATMs). Basically the concept of alternation is a generalization of nondeterminism in a way that allows existential and universal quantifiers to alternate during the course of a computation. Four kinds of states exist in an ATM; namely existential, universal, accepting and rejecting states. A universal state can lead to acceptance iff all successors lead to acceptance. On the other hand, an existential state can lead to acceptance iff there exists a successor that leads to acceptance. During the course of an ATM computation, the computation segment between two consecutive alternations is called an *alternation block*. The complexity classes of languages accepted by space (time) bounded ATMs were also defined in [2]. In particular, $A\Sigma_k^L$ ($A\Pi_k^L$) is the set of langauges accepted by O(log n) space bounded ATMs in which the starting state is an existential (universal) state and the machine is constrainted to make at most k-1 alternations during the course of a computation. (Any computation path of such a machine will have at most k alternation blocks.) Recently, it was shown by Lange, Jenner and Kirsig [12] that the alternating logspace hierarchy collapses to its second level, i.e., $A\Sigma_k^L = A\Pi_k^L = A\Sigma_2^L$, for all $k \geq 2$. Furthermore, we have: (see [2])

Theorem 2.1: NL, Co-NL $\subseteq D^L \subseteq A\Sigma_2^L \subseteq P \cap \mathrm{DSPACE}(\log^2 n)$.

3 The Main Results

In this section, we analyze the complexity results for the aforementioned problems. The results are summarized in the following table [2]:

	Existence	*Exact*	*Critical*	*Unique*
Cut point	D^L-C	–	D^L-C	$D^L..A\Sigma_2^L$
Knot	co-NL-C	$A\Sigma_2^L..P$	D^L-C	$A\Sigma_2^L$-C
Labelled knot	P-C	P-C	P-C	P-C

In a number of our lower bound proofs, we use a well-known NL-complete problem, namely, the *graph accessibility problem* (GAP), which is the problem of determining, from a given directed graph G and two vertices u and v, whether a directed path exists from u to v in G [8].

---

[2] In this table, X-C represents X-complete and Y..Z indicates that the lower bound and the upper bound are Y and Z, respectively.

In what follows, we first show the results concerning the existence of cut points, knots, and labelled knots.

Theorem 3.1: E-CUTP is D^L-complete.

Proof sketch. Let (G,x,y) be an instance of E-CUTP. First we show that the problem is in D^L. Let $A=\{(G,x,y) \mid x \rightsquigarrow y \text{ in } G\}$ and $B=\{(G,x,y) \mid x \not\rightsquigarrow y \text{ in } G$ if a vertex other than x and y is removed$\}$. Clearly, $A \in NL$ and $B \in$ Co-NL. Furthermore, $(G,x,y) \in$ E-CUTP iff $(G,x,y) \in A \cap B$. Therefore, E-CUTP is in D^L.

Now we show E-CUTP to be D^L-hard. Let L be an arbitrary language in D^L. Then $L=L_1 \cap L_2$, where $L_1 \in NL$ and $L_2 \in$ Co-NL. Let M_1 and M_2 be NTMs using logarithmic space such that $L(M_1)=L_1$ and $L(M_2)=L_2$. Given an input x, let G_1 and G_2 be the computation graphs of M_1 and M_2 (on x), respectively. Furthermore, without loss of generality, assume that G_1 and G_2 have disjoint vertex sets and each has exactly one starting and one accepting configuration. Let x_1 and y_1 (x_2 and y_2) be the starting and accepting configurations of G_1 (G_2), respectively. Let G_1' and G_2' be copies of G_1 and G_2, respectively. Let x_1' and y_1' (x_2' and y_2') be the starting and accepting configurations of G_1' (G_2'), respectively. (Note here that G_1, G_2, G_1' and G_2' are vertex disjoint.) Then construct a new graph G from G_1, G_2, G_1' and G_2' by

- adding the edge (y_1, x_2) and (y_1', x_2'),
- adding a new vertex x and edges (x, x_1) and (x, x_1'),
- adding a new vertex y and edges (y_2, y) and (y_2', y),
- adding a new vertex c and edges (x_2, c), (x_2', c) and (c,y).

It is not hard to see that $x \in L$ iff there exists a cut point (c) in G with respect to x and y. This completes the proof. ♣

Theorem 3.2: E-KNOT is Co-NL complete.

Proof sketch. Consider the complement of E-KNOT, namely, "given a graph G, does G not contain a knot?" It is not hard to see that G does not have a knot iff for every vertex v in G, v can reach some "deadend" vertex (i.e., a vertex with no successors) – perhaps itself. Since GAP is NL-complete, one can determine whether a knot exists by generating the vertices in some canonical order and then testing whether each generated vertex can reach a deadend vertex. As a result, E-KNOT is in Co-NL. The lower bound can be easily derived by reducing GAP to the complement of E-KNOT. ♣

Theorem 3.3: E-LKNOT is P-complete.

Proof sketch. A similar proof can be found in [18]. Here we just sketch the basic idea. To show that the problem is in P, we first decompose the graph into a set of maximum strongly connected components (SCCs). Then we test whether any of the SCCs is a labelled knot.

If so, the procedure terminates and returens "yes"; otherwise, a refinement of the SCCs is made to produce a new set of SCCs and then the procedure repeats. The lower bound can be derived by reducing the path system problem (see [9]), which is known to be P-complete, to E-LKNOT. ♣

The next two theorems deal with the exact versions of knots and labelled knots.

Theorem 3.4: X-KNOT is in P and $A\Sigma_2^L$-hard.

Proof sketch. The proof of the upper bound is similar to the proof of Theorem 3.5; proof of the lower bound is similar to the proof of Theorem 3.10. At this moment, we are unable to close the gap between the lower and the upper bounds. ♣

Theorem 3.5: X-LKNOT is P-complete.

Proof sketch. In the proof of Theorem 3.3, we first decompose G into a set of maximum SCCs. We then check to see whether a labelled knot exists in the set. If so, the procedure terminates successfully; otherwise, the set is refined and the procedure is repeated. To compute the maximum labelled knot, the procedure is modified so that in each step, the size of each SCC is also recorded. Furthermore, the procedure will continue (even when a labelled knot has been detected) until no more labelled knots exist. One can easily see that the modified procedure remains in P. The lower bound can be derived by reducing the path system to X-LKNOT. ♣

Theorems 3.6-3.8 concern themselves with the critical versions of cut points, knots and labelled knots.

Theorem 3.6: C-CUTP is D^L-complete.

Proof sketch. The proof is analogous to the proof of Theorem 3.1. ♣

Thoerem 3.7: C-KNOT is D^L-complete.

Proof sketch. The proof is similar in style to the proof of Theorem 3.6. ♣

Theorem 3.8: C-LKNOT is P-complete.

Proof sketch. Similar to the proof of Theorem 3.3. ♣

The next three theorems deal with the uniqueness problems.

Theorem 3.9: U-CUTP is in $A\Sigma_2^L$ and D^L-hard.

Proof sketch. Let (G,x,y) be an instance of U-CUTP. To show that U-CUTP is in $A\Sigma_2^L$, we first check whether a cut point exists (which can be done in D^L). We then nondeterministically guess two distinct vertices s and t and check whether s and t are cut points. If so, the cut point is not unique. It is reasonably easy to see that the above procedure can be performed in $A\Sigma_2^L$. The lower bound proof is similar to the proof of Theorem 3.1. ♣

Theorem 3.10: U-KNOT is $A\Sigma_2^L$-complete.

Proof. We first show U-KNOT to be in $A\Sigma_2^L$. Consider an arbitrary graph G. It is easy to see that G has a unique knot iff

1. $\exists$ a vertex u in G such that u can not reach any deadend vertex (i.e., u is in some knot), and

2. for every v in G, either u $\rightsquigarrow$ v (i.e., u and v are in the same knot) or v can reach some deadend vertex (i.e., v is not in a knot).

It is then not hard to see that U-KNOT $\in A\Sigma_3^L$. Due to a recent result that the alternation logspace hierarchy collapses [12], the upper bound follows.

The lower bound is derived by a "generic" reduction from $A\Sigma_2^L$ languages to U-KNOT instances. Let M be a 2-alternation O(log n) space bounded ATM accepting a language in $A\Sigma_2^L$. Without loss of generality, we assume that M makes an alternation along each computation path and that each computation path leads to either an accepting or rejecting configuration. On input x, each configuration of M is a 3-tuple (q,i,x), where q is a state, i is the input head position, and x represents the tape contents (including the tape head position). A configuration c=(q,i,x) is an existential (universal, accepting and rejecting, respectively) configuration iff q is an existential (universal, accepting and rejecting, respectively) state. Let S be the set of all such configurations. Let c_0 be the initial configuration. A graph G=(V,E) is constructed as follows:

1. $V = S \cup S \times S \times \{L,R\}$,
2.
   - (p,q) $\in$ E if p is an existential configuration and p can reach q in one step,
   - (r,[r,r,L]) and (r,[r,r,R]) are in E for every universal configuration r,
   - $\forall$ r $\in$ S, ([p,r,L],[q,r,L]) and ([p,r,R],[q,r,R]) $\in$ E if p is a universal configuration and p can reach q in one step,
   - (e,c_0) $\in$ E if e is an existential configuration,
   - ([u,r,L],[r,r,L]) and ([u,r,R],[r,r,R]) are in E if u and r are universal configurations,
   - ([a,r,L],c_0) and ([a,r,R],c_0) are in E if a in a rejecting configuration.

According to the construction, one can observe that any knot containing an existential configuration must also contain the starting configuration c_0. Now we claim that M accepts x iff G does not have a unique knot. To see this, first note that M accepts x iff there exists a path that leads to some universal configuration u, and from u no rejecting configurations are reachable. Now, in G two copies of the universal phase starting at u, say G_1 and G_2 exist – the one whose root is [u,u,L] and [u,u,R]. Now, if from u, the computation can reach only accepting configurations (i.e., M accepts x), then both G_1 and G_2 contain knots. In this case, G does not have a unique knot. On the other hand, if the computation can reach some rejecting configuration, then a knot involving G_1 and G_2 must also contain c_0. Therefore, it is not hard to see that G has a unique knot (the one involving c_0) iff M does not accept x. From [12], we have that $A\Sigma_2^L$ is closed under complementation, The lower bound follows. ♣

Theorem 3.11: U-CKNOT is P-complete.

Proof sketch. Similar to the proof of Theorem 3.5. ♣

4 Concluding remarks

The study of space complexity classes that are "below" PTIME has recently received broad attention. In Ruzzo, Simon and Tompa [16], the logspace oracle hierarchy which falls "between" NSPACE(log n) and PTIME $\cap$ DSPACE($log^2 n$) was defined. Subsequently, Rosier and Yen [17] defined another oracle hierarchy by restricting computations to allow at most one oracle call during the course of a computation. The resulting hierarchy – called the restricted logspace oracle hierarchy – was then shown to coincide with the alternating logspace hierarchy $A\Sigma_k^L$, $k \geq 0$, of [2]. Recently, Lange, Jenner, and Kirsig showed the surprising result that the alternating logspace hierarchy collapses to its second level, i.e., $A\Sigma_k^L = A\Sigma_2^L$, $k \geq 2$. After our paper was written, a somewhat more surprising result was shown by Immerman [7]: That for all space-constructable functions s(n) $\geq$ $logn$, NSPACE(s(n))=Co-NSPACE(s(n)). As a result, the alternating logspace hierarchy collapses all the way to NL. Due to this result, Theorem 2.1 can be upgraded to "$NL = Co\!-\!NL = D^L = A\Sigma_2^L \subseteq PTIME \cap DSPACE(log^2 n)$" and the D^L and $A\Sigma_2^L$ lower bound proofs of Section 3 are now trivial to show. The table of Section 3 should now be:

	Existence	*Exact*	*Critical*	*Unique*
Cut point	NL-C	–	NL-C	NL-C
Knot	NL-C	NL..P	NL-C	NL-C
Labelled knot	P-C	P-C	P-C	P-C

References

[1] Bentley, J., Ottmann, T. and Widmayer, P. *The complexity of manipulating hierarchically defined sets of rectangles.* Advances in Computing Research 1, JAI Press Inc., pp. 127-158, 1983.

[2] Chandra, A., Kozen, D. and Stockmeyer, L. Alternation. *JACM*, 28 (1) pp. 114–133, January 1981.

[3] S. Cook. The complexity of theorem proving procedures. *Proc. of the 3rd Annual ACM Symp. on Theory of Computing*, pp. 151–158, 1971.

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[5] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 1979.

[6] R. Howell, L. Rosier and H. Yen. Unary minimum cost problems, alternating logspace, and Ruzzo, Simon, and Tompa's DL^{NL}. Dept. of Computer Science, Iowa State University, TR 87-6, 1987.

[7] N. Immerman. Nondeterministic space is closed under completent. Tech. Rep., Computer Science Dept., Yale University, July 21, 1987.

[8] N. Jones. Space-bounded reducibility among combinatorial problems. *J. of Computer and System Sciences*, 11, pp. 68–75, 1975.

[9] N. Jones and W. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3, pp. 105–117, 1977.

[10] R. Karp. *Reducibility among combinatorial problems.* Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press,, New York, pp. 85-104, 1972.

[11] K. Lange. Two characterizations of the logarithmic alternation hierarchy. *Proc. of the 12th Symp. of Math. Foundations of Computer Science.* 1986, pp. 518-526.

[12] K. Lange, B. Jenner and B. Kirsig. The logarithmic alternation hierarchy collapse: $A\Sigma_2^L = A\Pi_2^L$. To be presented at the *14th International Colloquium on Automata, Languages, and Programming*, July, 1987, Karlsruhe, F.R.G.

[13] J. Misra and K. M. Chandy. A distributed graph algorithm: knot detection. *ACM Trans. on Programming Languages and Systems*, 4 (4) pp. 678–686, 1982.

[14] C. Papadimitriou. On the complexity of unique solutions. *JACM*, 31 (2) pp. 392–400, April 1984.

[15] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *J. of Computer and System Sciences*, 28 (2) pp. 244–259, 1984.

[16] W. Ruzzo, J. Simon and M. Tompa. Space-bounded hierarchies and probabilistic computations. *J. of Computer and System Sciences*, 28, pp. 216–230, 1984.

[17] L. Rosier and H. Yen. Logspace hierarchies, polynomial time and the complexity of fairness problems concerning ω-machines. *Proc. of 3rd Annual Symp. on Theoretical Aspects of Computer Science* (LNCS 210), pp. 306–320, 1986. To appear in *SIAM J. on Computing.*

[18] L. Rosier and H. Yen. On the complexity of deciding fair termination of probabilistic concurrent finite-state programs. *Proc. of 13th International Colloquium on Automata, Languages and Programming* (LNCS 226), pp. 334–343, 1986. To appear in *Theoretical Computer Science.*

COMPLEXITY AND DECIDABILITY FOR RESTRICTED CLASSES OF PICTURE LANGUAGES

CHANGWOOK KIM
School of Electrical Engineering and Computer Science
The University of Oklahoma
Norman, Oklahoma 73019

ABSTRACT

The paper investigates complexity and decidability questions for two restricted classes of picture languages, called three-way picture languages and stripe picture languages. It is shown that (1) the picture membership problem can be solved in deterministic polynomial time for three-way context-free picture languages, (2) the equivalence and containment problems for picture languages are undecidable for a regular language L_1 and a deterministic linear language L_2, where both L_1 and L_2 describe three-way stripe picture languages, (3) the intersection emptiness problem for picture languages is undecidable for a regular language L_1 and a deterministic context-free language L_2, where both L_1 and L_2 describe stripe picture languages, and (4) the intersection emptiness problem for picture languages is undecidable for two three-way stripe deterministic linear picture languages.

1. INTRODUCTION

Syntactic methods are one of the main approaches to pictorial pattern recognition and analysis [6]. Various models of grammars and automata have been developed to handle pictures, see, e.g., [11]. Among others, Freeman's "chain codes" and its variations have been used most popularly to encode pictures in these models [5]. Recently, the complexity of chain-encoded picture languages has been studied intensively [3, 8, 9, 10, 12], where a picture is described by a string over the alphabet $\{u, d, r, l\}$ with the following interpretation: The symbol u (d, r, and l, respectively) means "draw one unit line in the two-dimensional Cartesian plane by moving the pen up (down, right, and left, respectively) from the current point". The complexity is then considered using the standard Chomsky hierarchy of language families: regular, linear, context-free, context-sensitive, and recursively enumerable [7]. The result of these studies have shown that most of the important decision problems for picture languages are intractable. E.g., the membership problem is NP-complete for regular picture languages [12] and for context-free picture languages [8], the equivalence and containment problems are undecidable (not even partially decidable) for regular picture languages [8], and the intersection emptiness problem is not partially decidable for regular picture languages [12].

In real applications, however, we need not handle all possible picture encodings. Pictures for computer processing are usually highly domain specific, i.e., constrained by the unique properties of the problems at hand. This fact led to the investigation of restricted subclasses of picture languages which have better complexity results, while still possessing sufficient power for applications. In this regard, Sudborough and Welzl [12] considered the so-called stripe picture languages, whose pictures fit into a stripe defined by two parallel lines. Among others, they showed that, for regular stripe picture languages, the equivalence and intersection emptiness problems are decidable and the membership problem is linear time solvable.

In this paper, we consider another restricted class of picture languages, called three-way picture languages, whose description languages use only three letters. Informally, one can imagine that on a string over a three-letter alphabet the picture drawing device moves in only one direction on one axis, while moving freely on the other axis. Of course, a three-way device does not have enough power to handle general pictures. Despite that, we study this class since (1) it is a natural restriction to impose on a picture description model, (2) similar picture recognition devices, called three-way acceptors, have been introduced in an earlier publication [11], (3) there are problem domains which possess inherently three-way properties, e.g., electronic waveform processings [1, 2, 4], and (4) if a parallel processing system is used for picture recognition problems, a picture could be decomposed so that each single processor need only handle a three-way picture component.

We shall assume the reader to be familiar with the basics of formal language and automata theory [7]. For picture part, we informally discuss necessary notations, refering to [9] and [12] for more detailed definitions. We shall denote the length of a string x by $|x|$ and the empty string by λ. For a set A, 2^A denotes the power set of A and $|A|$ denotes the cardinality of A.

Let $\pi = \{u, d, r, l\}$ be an alphabet. Each string w in π^* describes a *picture* in the sense that it describes a walk in the two-dimensional Cartesian plane, where u, d, r, and l represent movements up, down, right, and left, respectively. For a string $w \in \pi^*$, the *drawn picture described by w*, denoted by $\text{dpic}(w)$, is a triple $\langle p, s, e\rangle$, where p is a picture, s is the *start point*, and e is the *end point*. If $\text{dpic}(w) = \langle p, s, e\rangle$, then the *basic picture described by w*, denoted by $\text{bpic}(w)$, is p.

A drawn or basic picture is a *stripe picture* if it fits into a stripe defined by two parallel lines on the plane. See Fig. 1 for an example (a little circle or square in a picture denotes the start or end point, respectively). A set of pictures P is a *stripe picture language* if every picture in P fits into a fixed stripe.

Consider a three-letter subset $\tilde{\pi}$ of π. A picture is called a *three-way picture* if it can be described by a $\tilde{\pi}$-word (see Fig. 2 for an example) and a picture language is called a *three-way picture language* if it can be described by a $\tilde{\pi}$-language. For the discussions that follow, we shall fix $\tilde{\pi}$ to be $\{u, d, r\}$.

For a language L over π, the *set of drawn pictures described by L*, denoted by $\text{dpic}(L)$, is $\text{dpic}(L) = \{\text{dpic} \mid w \in L\}$. Similarly, the *set of basic pictures described by L*, denoted by $\text{bpic}(L)$, is $\text{bpic}(L) = \{\text{bpic}(w) \mid w \in L\}$. A set of drawn (or basic) pictures P is *regular, linear*, or *context-free* if there is a regular, linear, or context-free language L such that $\text{dpic}(L) = P$ ($\text{bpic}(L) = P$, respectively).

2. MEMBERSHIP COMPLEXITY FOR THREE-WAY PICTURE LANGUAGES

A three-way picture can be described by a sequence of line-disjoint horizontal and vertical line segments which come in turn. For example, the drawn picture q in Fig. 2.(a) can be partitioned into 9 subpictures $q_1, q_2, \ldots, q_9$, as described in Fig. 2.(b) from left to right, so that $q = q_1 \cdot q_2 \cdots q_9$. Every $\tilde{\pi}$-word that describes q must draw q in this sequence. We call the partition of a three-way picture p into such a sequence of line-disjoint horizontal and vertical line segments the *(ordered) partition* of p and denote by $\text{part}(p)$. If $p_1, p_2, \ldots, p_m$ is such a sequence for p, we write $\text{part}(p) = (p_1, p_2, \ldots, p_m)$, where, for all $i = 1, 2, \ldots, m$, p_i is a drawn (basic, respectively) picture if p is. By convention, $p_i \neq \emptyset$ for all i if $p \neq \emptyset$ ($\text{part}(p) = \emptyset$ if $p = \emptyset$).

Let p be a drawn picture and $M = (\{1, 2, \ldots, t\}, \tilde{\pi}, \delta, 1, t)$ be a finite automaton. We want to determine whether or not $p \in \text{dpic}(M)$. Let $\text{part}(p) = (p_1, p_2, \ldots, p_m)$

for some integer m. Without loss of generality we assume that p_1 is a horizontal line segment and m is an even number. For $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, t$, construct new finite automata $M^v_{i,j}$ and $M^h_{i,j}$ as follows:

$$M^v_{i,j} = (Q, \tilde{\pi}, \delta^v_{i,j}, i, j), \text{ where } \delta^v_{i,j}(q,a) = \begin{cases} \delta(q,a) & \text{if } a = u \text{ or } d, \\ \emptyset & \text{if } a = r, \end{cases}$$

$$M^h_{i,j} = (Q, \tilde{\pi}, \delta^h_{i,j}, i, j), \text{ where } \delta^h_{i,j}(q,a) = \begin{cases} \delta(q,a) & \text{if } a = r, \\ \emptyset & \text{if } a = u \text{ or } d. \end{cases}$$

For $k = 1, 2, \ldots, m$, let T_k be a $t \times t$ Boolean matrix defined by

$$T_k[i,j] = \begin{cases} 1 & \text{if } p_k \in \mathrm{dpic}(M^a_{i,j}), \\ 0 & \text{otherwise,} \end{cases}$$

where $a = h$ if k odd, and $a = v$ if k even. Let T be a $t \times t$ Boolean matrix defined by

$$T = \prod_{k=1}^{m} T_k.$$

LEMMA 2.1. *$p \in dpic(M)$ if and only if $T[1,t] = 1$.*

PROOF: Omitted. ♣

Lemma 2.1 yields the following algorithm for the membership problem of three-way regular picture languages.

ALGORITHM 2.1. *Membership Testing for Three-Way Regular Picture Languages.*
Input: A drawn picture $\langle p, s, e\rangle$ and a fixed FA $M = (\{1, 2, \ldots, t\}, \tilde{\pi}, \delta, 1, t)$.
Output: "yes" if $\langle p, s, e\rangle \in dpic(M)$, "no" otherwise.
Method:
(1) Compute $part(p) = (p_1, p_2, \ldots, p_m)$.
(2) Construct $M^v_{i,j}$ and $M^h_{i,j}$ for all i and j, $1 \le i, j \le t$.
(3) Construct T_k for $k = 1, 2, \ldots, m$.
(4) Compute T.
(5) If $T[1,t] = 1$ then output "yes", else output "no".

It is not difficult to observe that Algorithm 2.1 takes $O(|p|^2)$ time and that with slight modification it works for basic pictures. Thus, we have the following theorem:

THEOREM 2.2. *The membership problem for drawn and basic three-way regular picture languages can be solved in $O(|p|^2)$ time deterministically.*

Let $M = (Q, \tilde{\pi}, \Gamma, \delta, q_0, Z_0, f)$ be a PDA and $\langle p, s, e\rangle$ be a drawn picture. We want to determine whether or not $\langle p, s, e\rangle \in \mathrm{dpic}(M)$. We first define some notions regarding the picture drawing behaviors of PDA M. A *configuration* of M is a 5-tuple (q, w, p', v, v'), where q is a state of M, w is a (possibly empty) string of pushdown store symbols of M, and $\langle p', v, v'\rangle$ is a drawn picture. The configuration (q, w, p', v, v') means that the PDA M is in state q, the current content of the pushdown store is w (with the first symbol of w being the top symbol of the pushdown store), and M has read some portion of the input tape, say x, such that $\mathrm{dpic}(x) = \langle p', v, v'\rangle$. If (q, w, p', v, v') is a configuration of M, then the 4-tuple (q, A, p', v') is called a *surface configuration* of M, where A is the first symbol of w ($A = \lambda$ if $w = \lambda$). Binary move relations $\vdash$, $\vdash^*$, $\vdash^+$, $\vdash^k$ on configurations and surface configurations of M are defined as usual.

Let C be the set of all 6-tuples in $Q \times \Gamma \times Q \times 2^{M_1} \times M_0 \times M_0$. A 6-tuple (q, A, r, p', v, v') in C is called *p-valid* if the PDA M can go from state q with symbol A alone on the pushdown store to state r with an empty store, drawing the picture $p' \subseteq p$, starting from $v \in M_0$ and ending at $v' \in M_0$, i.e., if $(q, A, \emptyset, v) \vdash^* (r, \emptyset, p', v')$. Let $V(p) = \{v_1(= s), v_2, \ldots, v_m(= e)\}$ and X be an $m \times m$ matrix such that, for all i and j, $1 \le i, j \le m$, the entry $X[i,j]$ contains the set

$$\{(q, A, r, p') \mid q, r \in Q, A \in \Gamma, \text{ and } (q, A, r, p', v_i, v_j) \text{ is a } p\text{-valid 6-tuple}\}.$$

LEMMA 2.3. *$\langle p, s, e\rangle \in dpic(M)$ if and only if $X[1,m]$ contains (q_0, Z_0, f, p).*

PROOF: The result follows directly from the definition of p-valid 6-tuples. ♣

It follows from Lemma 2.3 that we need only construct the matrix X described above in order to test the membership of three-way context-free picture languages. We show in the following that X can be constructed in polynomial time deterministically. To this end, we use a dynamic programming method which (1) puts in the table X all p-valid 6-tuples corresponding to single pop transitions of M, (2) combines repeatedly the informations in the table X that correspond to partial computations of M until no new information is added to X, and (3) checks whether or not the 4-tuple (q_0, Z_0, f, p) has been added to $X[1,m]$. Dynamic programming method used for constructing the matrix X can be suitably defined by a function $\text{YIELD}_M : (2^C)^2 \to 2^C$ as follows:

$$\begin{aligned}\text{YIELD}_M(S_1, S_2) = \{(q, B, t, p', v, v''') \mid{} & (q, B, \emptyset, v) \vdash (r, AB, p_1, v'),\\ & (r, A, s, p_2, v', v'') \in S_1,\ (s, B, t, p_3, v'', v''') \in S_2,\\ & \text{and } p' = p_1 \cup p_2 \cup p_3\}.\end{aligned}$$

ALGORITHM 2.2. *Membership Testing for Three-Way Context-Free Picture Languages.*

Input: A drawn picture $\langle p, s, e\rangle$ and a fixed PDA $M = (Q, \tilde{\pi}, \Gamma, \delta, q_0, Z_0, f)$.

Output: "yes" if $\langle p, s, e\rangle \in dpic(M)$, and "no" otherwise.

Method:

```
(1) If $\langle p, s, e\rangle$ is not a three-way picture, then output "no".
(2) Set a stack $T$ to be empty, and set $X[i,j] \leftarrow \emptyset$, $1 \le i, j \le m$ $(= |V(p)|)$.
(3) Add to $T$ all p-valid 6-tuples, say $(q, A, r, p', v_i, v_j)$, corresponding to single pop
    transitions of $M$. Simultaneously, store $(q, A, r, p')$ in $X[i,j]$.
(4) while $T$ is not empty do call BUILDUP.
(5) if $X[1,m]$ contains $(q_0, Z_0, f, p)$ then output "yes" else output "no".

    procedure BUILDUP;
(a) Pop the top entry, say $top = (r, A, s, p', v_i, v_j)$, of the stack $T$;
(b) /* right build-up */
    for $k = 1$ to $m$ do
        Let $S$ be the set of 6-tuples corresponding to $X[j,k]$;
        Compute $E = YIELD_M(\{top\}, S)$;
        for each 6-tuple, say $Y = (q, B, t, p', v_l, v_k)$, in $E$ if $p'$ is a subpicture of $p$ and
        $Y$ is not in $X$ then add $(q, B, t, p')$ to $X[l,k]$ and put $Y$ in $T$;
    endfor;
```

```
(c) /* left-buildup */
    for $k = 1$ to $m$ do
        Let $S$ be the set of 6-tuples corresponding to $X[k,i]$;
        Compute $E = YIELD_M(S, \{top\})$;
        for each 6-tuple, say $Y = (q, B, t, p', v_l, v_j)$, in $E$ if $p'$ is a subpicture of $p$ and
            $Y$ is not in $X$ then add $(q, B, t, p')$ to $X[l, j]$ and $Y$ in $T$;
    endfor;
    endprocedure.
```

Lemma 2.4. *Every p-valid 6-tuple gets added to the table X in Algorithm 2.2.*

Proof: Omitted. ♣

Lemma 2.5. *Algorithm 2.2 terminates within* $O(|p|^{12})$ *time deterministically.*

Proof: Omitted. ♣

To test the membership of a basic picture p, we can just run the same algorithm and check if $X[i,j]$ contains the element (q_0, Z_0, f, p) for any i and j. This yields the same time requirement for the membership testing of basic three-way context-free picture languages. Therefore, we have the following theorem:

Theorem 2.6. *The membership problem for drawn and basic three-way context-free picture languages can be solved in* $O(|p|^{12})$ *time deterministically.*

3. Undecidability Results

Theorem 3.1. *It is not partially decidable whether or not (1)* $dpic(L_1) = dpic(L_2)$*, and (2)* $bpic(L_1) = bpic(L_2)$*, for a regular* $\tilde{\pi}$*-language* L_1 *and a deterministic linear* $\tilde{\pi}$*-language* L_2*, where both* $bpic(L_1)$ *and* $bpic(L_2)$ *are stripe picture languages.*

Proof: It is straightforward to observe that the complement problem, i.e., the inequivalence problem is partially decidable. To complete the proof, we show that the inequivalence problem is undecidable. This will be done by a reduction from PCP.

Let $I = (x, y)$ be an arbitrary instance of PCP over the alphabet $\Sigma = \{a, b\}$, where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. Define two homomorphisms h_x and h_y from Σ^* to $\tilde{\pi}^*$ by the following conditions: $h_x(a) = udr$, $h_x(b) = u^2d^2r$, $h_y(a) = rud$, and $h_y(b) = ru^2d^2$.

Construct a right-linear grammar $G = (N, \tilde{\pi}, P, X)$ with nonterminals $N = (X, Y)$ and the following productions:

$$X \rightarrow d^i u^i\ h_x(x_i)\ X \mid d^i u^i\ h_x(x_i)\ urd\ Y,$$
$$Y \rightarrow h_y({y_i}^R)\ d^i u^i\ Y \mid h_y({y_i}^R)\ d^i u^i,$$

for all $i = 1, 2, \ldots, n$.

Intuitively speaking, a word generated by G draws a picture of the form described in Fig. 3, where the "thorns" drawn above the left horizontal line segment of the single bridge (the bridge is the picture component described by string udr) simulate a word obtained by concatenating x_i's and the "thorns" drawn below the left horizontal line segment of the single bridge simulate the order of the indices of those x_i's concatenated. Similarly, the thorns drawn above (below, respectively) the right horizontal line segment of the bridge simulate a word obtained by concatenating y_j's (the order of the indices of those y_j's concatenated, respectively), but in the reversed order in this case. Thus,

$I = (x, y)$ has a solution to PCP if and only if there exists a picture p in dpic(G) such that the subpicture of p which lies above the two long horizontal line segments constitute a "picture palindrome" (i.e., symmetric with respect to the bridge) and the subpicture of p which lies below the two long horizontal line segments constitute a "picture palindrome in a weak sense" (i.e., the order of the occurrences of the thorns are symmetric with respect to the bridge). We call a picture p with the two properties stated above a *solution picture* for $I = (x, y)$.

For a technical reason, we define, for $i = 1$ and 2, right-linear grammars $G_i = (N_i, \tilde{\pi}, P_i, X_i)$ with $N_i = N \cup \{X_i\}$ and the productions $P_1 = P \cup \{X_1 \rightarrow du\ X\}$, and $P_2 = P \cup \{X_2 \rightarrow ud\ X\}$. It is not difficult to see that dpic(G_1) = dpic(G_2) = dpic(G). Let $w = dux$ ($z = udx$, respectively) be an arbitrary word generated by G_1 (G_2, respectively), where x is represented by:

$$x = d^{i_1} u^{i_1}\ h_x(x_{i_1}) \cdots d^{i_p} u^{i_p}\ h_x(x_{i_p})\ urd\ h_y({y_{j_q}}^R)\ d^{j_q} u^{j_q} \cdots h_y({y_{j_1}}^R)\ d^{j_1} u^{j_1}.$$

We can construct one-way pushdown automata A_1 and A_2 which accept subsets of $L(G_1)$ and $L(G_2)$, respectively, in the following manner: A_1 accepts w if and only if $w \in L(G_1)$ and $(i_1, i_2, \ldots, i_p) \neq (j_1, j_2, \ldots, j_q)$, and A_2 accepts z if and only if $z \in L(G_2)$ and $(x_{i_1} x_{i_2} \cdots x_{i_p})^R \neq ({y_{j_q}}^R {y_{j_{q-1}}}^R \cdots {y_{j_1}}^R)$, i.e., $x_{i_1} x_{i_2} \cdots x_{i_p} \neq y_{j_1} y_{j_2} \cdots y_{j_q}$.

Observe that the automata A_1 and A_2 are one-turn pushdown automata, i.e., pushdown automata which, once decrease the size of its pushdown store, never increase it again during the whole computation on any input [7]. Furthermore, A_1 and A_2 are both deterministic automata. Therefore, $L_1 = L(G_1)$ and $L_2 = L(G_2)$ are both deterministic linear languages. Observe that $L = L_1 \cup L_2$ is also a deterministic linear language: A one-way deterministic pushdown automaton A can be easily constructed which first reads the leftmost input symbol from the tape and then simulates either A_1 or A_2 according to whether the symbol read is d or u, respectively.

From our construction of A_1 and A_2, it is clear that the automaton A never accepts a word w which describes a solution picture and that dpic(G) − dpic(A) can only contain solution pictures. It follows that $I = (x, y)$ has a solution to PCP if and only if dpic(G) − dpic(A) $\neq \emptyset$ if and only if dpic(G) $\neq$ dpic(A).

The proof described above holds equally for bpic(G) and bpic(A) since for every picture p in bpic(G) (bpic(A), respectively) there is a unique word w in $L(G)$ ($L(A)$, respectively) such that bpic(w) = p. It is obvious that bpic(G) and bpic(A) are both stripe picture languages. Hence, the theorem follows. ♣

THEOREM 3.2. *It is not partially decidable whether or not (1) dpic(L_1) $\subseteq$ dpic(L_2), and (2) bpic(L_1) $\subseteq$ bpic(L_2), for a regular $\tilde{\pi}$-language L_1 and a deterministic linear $\tilde{\pi}$-language L_2, where both bpic(L_1) and bpic(L_2) are stripe picture languages.*

PROOF: In the proof of Theorem 3.1, $I = (x, y)$ has a solution to PCP if and only if dpic(G) is not a subset of dpic(A) if and only if bpic(G) is not a subset of bpic(A). So, the same reduction proves the current theorem. ♣

THEOREM 3.3. *It is not partially decidable whether or not (1) dpic(L_1) $\cap$ dpic(L_2) = $\emptyset$, and (2) bpic(L_1) $\cap$ bpic(L_2) = $\emptyset$, for a regular π-language L_1 and a deterministic context-free π-language L_2, where both bpic(L_1) and bpic(L_2) are stripe picture languages.*

PROOF: Let $I = (x, y)$ be an arbitrary instance of PCP over the alphabet $\Sigma = \{a, b\}$, where $x = (x_1, x_2, \ldots, x_n)$, and $y = (y_1, y_2, \ldots, y_n)$. We construct a right-linear grammar G and a deterministic pushdown automaton A such that $I = (x, y)$ has a solution to PCP if and only if $\text{dpic}(G) \cap \text{dpic}(A) \neq \emptyset$. Since the problem of determining whether or not $\text{dpic}(G) \cap \text{dpic}(A) \neq \emptyset$ is partially decidable, it implies that the intersection emptiness problem is not partially decidable.

Let h_x and h_y be homomorphisms from Σ^* to π^* defined by $h_x(a) = u^2rdldr^3$, $h_x(b) = u^3rd^2ldr^3$, $h_y(a) = r^3u^2rdld$, and $h_y(b) = r^3u^3rd^2ld$. Homomorphisms h_x and h_y map the symbols in Σ to the pictures described in Fig. 4. The following picture components are defined to describe the grammar G and the automaton A: $\langle\text{start}\rangle = urdl$, $\langle\text{bridge}\rangle = u^2rd^2l$, and $\langle\text{index}\rangle_i = d^ilr^2lu^i$ for $i = 1, 2, \ldots, n$.

Let $G = (N, \pi, P, S)$ be a right-linear grammar with $N = \{S, X, Y, Z\}$ and the productions P as follows:

$$\begin{aligned}
&S \to \langle\text{start}\rangle\, r^3\, X,\\
&X \to \langle\text{index}\rangle_i\; h_x(x_i)\; X \quad \text{for } i = 1, 2, \ldots, n,\\
&X \to \langle\text{bridge}\rangle\; Y,\\
&Y \to h_y({y_i}^R)\; \langle\text{index}\rangle_i\; Y \quad \text{for } i = 1, 2, \ldots, n,\\
&Y \to l^3\; Z,\\
&Z \to l^3\; Z \mid \lambda.
\end{aligned}$$

Let $L_1 = L(G)$. The idea behind the construction of the grammar G is similar to the one in the proof of Theorem 3.1 except that we use slightly different picture components for simulating the symbols in Σ and the indices of the words concatenated. Intuitively speaking, a π-word generated by G draws a picture of the form described in Fig. 5, where the leftmost square indicates the marking of the start of a picture drawing, the subpicture drawn above the left horizontal line segment of the single bridge (the bridge is represented by a rectangle with height 2 and width 1 on the horizontal line segment) simulates a word obtained by concatenating x_i's, and the subpicture drawn below the left horizontal line segment of the single bridge simulates the order of the indices of those x_i's concatenated. Similarly, the subpicture drawn above (below, respectively) the right horizontal line segment of the single bridge simulates a word obtained by concatenating y_j's (the order of the indices of those y_j's concatenated, respectively), but in the reversed order in this case.

Observe that $I = (x, y)$ has a solution to PCP if and only if there exists a picture p in $\text{dpic}(G)$ such that (1) the subpictures of p corresponding to the symbols a and b appear symmetrically with respect to the bridge, and (2) the subpictures of p corresponding to the indices of x_i's and y_j's concatenated appear symmetrically with respect to the bridge. We call a picture satisfying these two conditions a *solution picture* for $I = (x, y)$.

Now we define the second π-language L_2, a deterministic context-free language, used in our reduction. Let g_x and g_y be homomorphisms from Σ^* to π^* defined by $g_x(a) = l^3u^2rdld$, $g_x(b) = l^3u^3rd^2ld$, $g_y(a) = u^2rdldl^3$, and $g_y(b) = u^3rd^2ldl^3$. Homomorphisms g_x and g_y map the symbols in Σ to the pictures described in Fig. 4, that are defined by homomorphisms h_x and h_y, with start points and end points exchanged. For $i = 1, 2, \ldots, n$, define the following π-words which describe two subpictures of the picture described by $\langle\text{index}\rangle_i$: $\langle\text{r-index}\rangle_i = d^irlu^i$, and $\langle\text{l-index}\rangle_i = d^ilru^i$, and construct a one-way pushdown automaton A that works as follows:

Phase 1. /* Symbol Comparison and Index Setting */
(1) Read $\langle\text{start}\rangle r^3$.

(2) Read one of $\langle$r-index$\rangle_i$, followed by $h_x(x_i)$, $1 \le i \le n$, pushing x_i onto the pushdown store. Repeat it an arbitrary number of times.
(3) Read $\langle$bridge$\rangle$.
(4) Read one of $h_y(y_i{}^R)$, followed by $\langle$r-index$\rangle_i$, $1 \le i \le n$, comparing the symbols in $y_i{}^R$ with those in the pushdown store. Repeat it an arbitrary number of times. Meanwhile, if any pair of symbols in $y_i{}^R$ and the pushdown store does not agree with each other, then stop without accepting.

Phase 2. /* Index Verification */
(1) Read one of $\langle$l-index$\rangle_i$, followed by $g_y(y_i)$, $1 \le i \le n$, pushing the index i onto the pushdown store. Repeat it an arbitrary number of times.
(2) Read $\langle$bridge$\rangle$.
(3) Read one of $g_x(x_i{}^R)$, followed by $\langle$l-index$\rangle_i$, $1 \le i \le n$, comparing the index i with the one on the top of pushdown store. Repeat it an arbitrary number of times. Meanwhile, if any pair of indices does not agree with each other, then stop without accepting.
(4) Read $l^3\langle$start$\rangle$.
(5) If the input head scans the right endmarker and the pushdown store is empty, then stop and accept.

Let $L_2 = L(A)$. In Phase 1, the automaton A draws a picture p of the form described in Fig. 6.(a) which is similar to a picture drawn by G except that (1) the subpictures of p corresponding to the symbols in Σ appear symmetrically with respect to the bridge, and (2) the subpictures of p corresponding to the indices of x_i's and y_j's concatenated each has a single "right thorn" at the bottom, whereas the pictures drawn by G has double thorns in the same place. In Phase 2, A draws a picture p' of the form described in Fig. 6.(b) which is similar to a picture drawn by G except that (1) the subpictures of p' corresponding to indices of x_i's and y_j's less the thorns at the bottoms appear symmetrically with respect to the bridge, and (2) these subpictures each has a single "left thorn" at the bottom. The automaton A draws p' by moving from right to left on the plane, performing a task similar to the one in Phase 1 in reversed order.

In other words, the automaton A draws a picture p in Phase 1 which satisfies the first condition for a picture to be a solution picture, and in Phase 2, A verifies that the indices drawn in Phase 1 satisfy the second condition for a picture to be a solution picture by drawing a picture p' which (1) guesses the picture components drawn in Phase 1, including the symbols in Σ, the locations of the indices, the bridge, and the start marker, and (2) verifies that the picture components drawn below the horizontal line segments in Phase 1, which correspond to the indices of x_i's and y_j's concatenated, appear symmetrically with respect to the bridge, using the complementary thorns.

From our construction of G and A, the reader can easily see that $I = (x, y)$ has a solution to PCP if and only if $\text{dpic}(G) \cap \text{dpic}(A) \neq \emptyset$. Also, it is not difficult to observe that A is actually a deterministic device, even though the picture drawing task in Phase 2 is conceptually nondeterministic. The argument given above equally holds for basic picture case, too. Since the intersection nonemptiness problem is partially decidable, the theorem follows. ♣

THEOREM 3.4. *It is not partially decidable whether or not (1) $dpic(L_1) \cap dpic(L_2) = \emptyset$, and (2) $bpic(L_1) \cap bpic(L_2) = \emptyset$, for two deterministic linear $\tilde{\pi}$-languages L_1 and L_2, where both $bpic(L_1)$ and $bpic(L_2)$ are stripe picture languages.*

PROOF: Recall the construction of one-turn pushdown automata A_1 and A_2 in the proof of Theorem 3.1. For the proof of the current theorem, we let A_1 accept a word w, described in the proof of Theorem 3.1, if and only if $w \in L(G_1)$ and $(i_1, i_2, \ldots, i_p) = (j_1, j_2, \ldots, j_q)$ and A_2 accept a word z, also described in the proof of Theorem 3.1, if and only if $z \in L(G_2)$ and $(x_{i_1} x_{i_2} \cdots x_{i_p})^R = ({y_{j_q}}^R {y_{j_{q-1}}}^R \cdots {y_{j_1}}^R)$, where inequalities have been replaced with equalities in both relations. An instance $I = (x, y)$ of PCP has a solution if and only if there exists a nonempty picture p in $\text{dpic}(A_1) \cap \text{dpic}(A_2)$, i.e., p is a solution picture described by two words $w \in L(A_1)$ and $z \in L(A_2)$, where w satisfies the first equation above and z satisfies the second equation above. The basic picture case is similar. ♣

ACKNOWLEDGEMENT

The author is grateful to Ivan Hal Sudborough for his encouragement and help during the study of this subject.

REFERENCES

[1] Belforte, G., R. De Mori, and F. Ferraris, A contribution to the automatic processing of electrocardiograms using syntactic methods, *IEEE Biomed. Eng., BME-26* (1979), 125-136.

[2] Birman, K. P., Rule-based learning for more accurate ECG analysis, *IEEE Pattern Analysis and Machine Intelligence, PAMI-4, No. 4* (1982), 369-380.

[3] Culik II, K., and E. Welzl, Two-way finite state generators, in *Foundations of Computation Theory* (M. Karpinsky, ed.), Lecture Notes in Computer Science 158, Springer, Berlin, 1983, 106-114.

[4] Dwyer, S. J., C. A. Harlow, D. A. Ausherman, and G. S. Lodwick, Computer diagnosis of radiographic images, *Proc. SJCC 40* (1972), 1027-1041.

[5] Freeman, H., On the encoding of arbitrary geometric configurations, *IRE Trans. EC-10* (1961), 260-268.

[6] Fu, K. S., *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[7] Hopcroft, J. E., and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979.

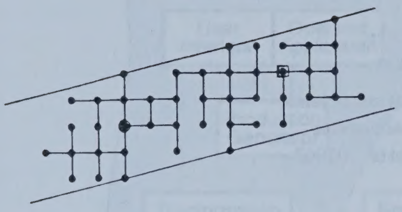
[8] Kim, C., and I. H. Sudborough, The membership and equivalence problems for picture languages, *Proc. 24th Annual Allerton Conf. on Communication, Control, and Computing* (1986), 730-739 (An extended version will appear in *Theoretical Computer Science 53* (1987), 1-15).

[9] Maurer, H. A., G. Rozenberg, and E. Welzl, Using string languages to describe picture languages, *Information and Control 54* (1982), 155-185.

[10] Maurer, H. A., G. Rozenberg, and E. Welzl, Chain code picture languages, in *Graph Grammars and Their Application to Computer Science* (H. Ehrig, N. Nagl, and G. Rozenberg, Eds.), Lecture Notes in Computer Science 153 (1983), 232-244.

[11] Rosenfeld, A., *Picture Languages - Formal Models for Picture Recognition*, Academic Press, New York, 1979.

[12] Sudborough, I. H., and E. Welzl, Complexity and decidability for chain code picture languages, *Theoretical Computer Science 36* (1985), 173-202.



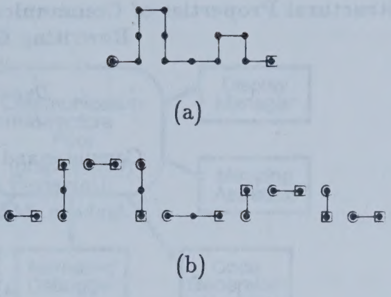


Fig. 1. A drawn stripe picture.

Fig. 2. A three-way picture and its partition.

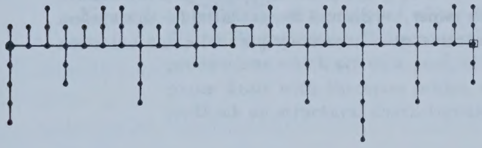


Fig. 3. An example of a picture in dpic(G).

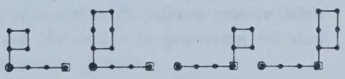


Fig. 4. Picture components for simulating alphabet symbols of PCP.

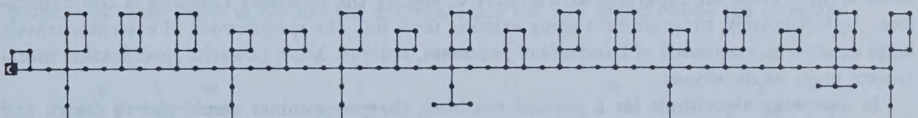


Fig. 5. An example of a picture in dpic(G).

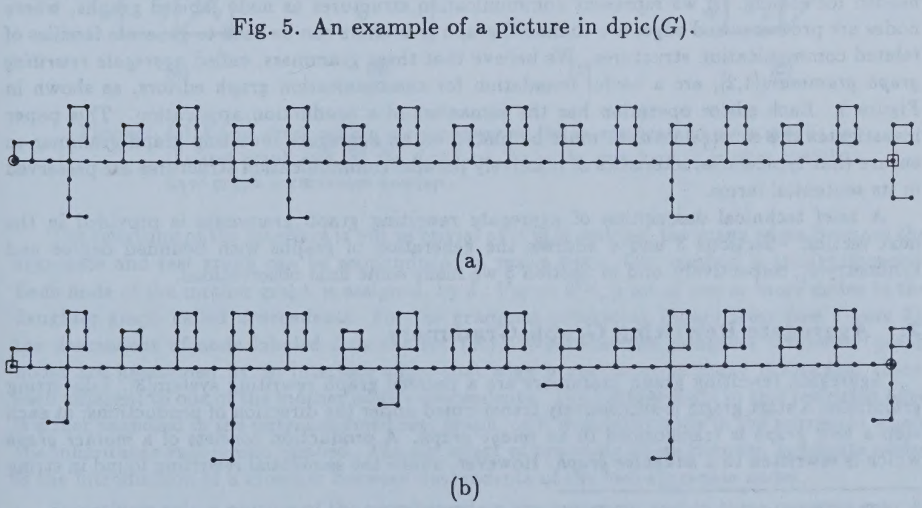


Fig. 6. An example of a picture in dpic(A).

Structural Properties of Communication Structures Generated by Aggregate Rewriting Graph Grammars*

Duane A. Bailey
Janice E. Cuny
Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

Efficient programming of massively parallel algorithms often requires that the structure of communication be explicitly specified. Concise description of the necessary channels of communication for thousands processes is a difficult task. We have proposed the use of a parallel graph-rewriting system – aggregate rewriting graph grammars[1,2] – as a mechanism for directing the specification process. In this paper, we discuss the potential for this system to describe graphs with the sparsity and symmetry of massively parallel communication structures.

1. Introduction

Architectures which support massive amounts of parallelism will, out of necessity, have restricted interprocess communication capabilities. In order to efficiently map process sets onto these architectures the programmer will have to specify the necessary channels of communication. Unfortunately, programmers using existing tools find the specification of even moderately large structures, composed of hundreds of processes, tedious. More powerful specification mechanisms must be developed.

In designing algorithms for a parallel machine, the programmer would like to design and test his program in-the-small, and then scale it to fit the currently available hardware. Unfortunately, existing tools force him to specify a single instance rather than the *family* of graphs needed for scaling. If we represent communication structures as node labeled graphs, where nodes are processes and edges are channels, graph grammars can be used to generate families of related communication structures. We believe that these grammars, called *aggregate rewriting graph grammars*[1,2], are a useful foundation for communication graph editors, as shown in Figure 1. Each editor operation has the semantics of a production application. This paper investigates the constraints that must be placed on an aggregate rewriting graph grammar to ensure that typical characteristics of massively parallel communication structures are preserved in its sentential forms.

A brief technical description of aggregate rewriting graph grammars is provided in the next section. Sections 3 and 4 address the generation of graphs with bounded degree and symmetry[5], respectively, and in Section 5 we make some final observations.

2. Aggregate Rewriting Graph Grammars

Aggregate rewriting graph grammars are a parallel graph rewriting system[3]. Like string grammars, a start graph is successively transformed under the direction of productions; at each step a *host graph* is transformed to an *image graph*. A production consists of a *mother graph* which is rewritten to a *daughter graph*. However, unlike the sequential rewriting found in string

---

*The Parallel Programming Environments Project at the University of Massachusetts is supported by the Office of Naval Research, contract N000014-84-K-0647.

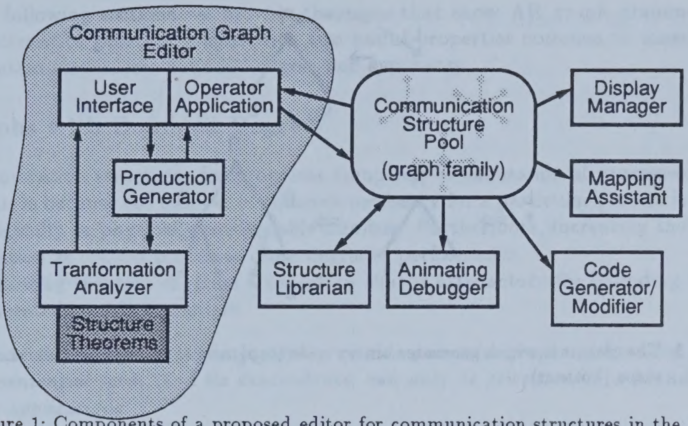

Communication Graph Editor
User Interface
Operator Application
Production Generator
Tranformation Analyzer
Structure Theorems
Communication Structure Pool (graph family)
Display Manager
Mapping Assistant
Structure Librarian
Animating Debugger
Code Generator/ Modifier

Figure 1: Components of a proposed editor for communication structures in the Simple Simon Programming Environment[4]. User requests become graph grammar productions which act on a pool, or family of structurally related graphs. This paper deals with theorems which would aid the editor in providing running feedback on structural characteristics of generated graphs.

grammars, *every* occurrence of the mother graph within the host graph is replaced, in parallel, by consistently labeled, disjoint daughter graphs. The collection of all occurrences of the mother graph in the host graph is an *aggregate* (see Figure 2). We identify these occurrences using bijections $\psi_i : M \rightarrow M_i \subseteq H$, indexed over set J. The host graph less the aggregate is the *rest graph*, and those edges in neither the aggregate nor the rest graph define the *interface*.

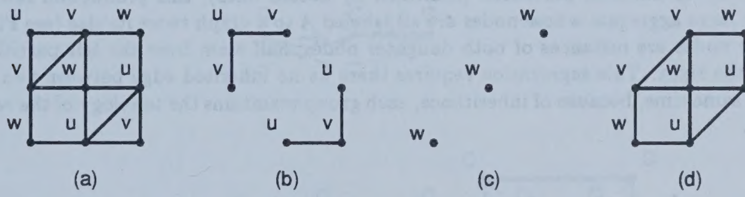

u v w
v w u
w u v
u v
v u
u v
w
w
w
v w
v w u
w u
(a)
(b)
(c)
(d)

Figure 2: (a) A host graph with $u - v$ occurrences, (b) an aggregate of $u - v$ graphs, (c) the rest graph and (d) the interface. As is the case here, it is often useful to have graph occurrences overlap.

The rewriting of the interface is not as trivial as it is in strings: the many edges between the aggregate and rest graph can be reintroduced in many ways. Our method is straightforward. Each node of the mother graph is assigned, by $\delta : V_M \rightarrow 2^{V_D}$, a set of one or more nodes in the daughter graph called *descendents*. For the grammar generating binary trees (see Figure 3), the descendent of node labeled L is the set $\{R\}$. Whenever the instances of mother graph nodes are mentioned by an interface edge, that edge is rewritten as many *descendent edges*, each adjacent to one of the mother node's descendents. The partner node in this rewritten edge is either anchored in the untransformed rest graph, or it is another node in the aggregate where the inheritance rule is also applied. The net effect of rewriting edges between aggregate nodes is the introduction of a crossbar between descendents of the two aggregate nodes.

Sometimes only a portion of the crossbar edges are necessary, and in these cases we may *k-partition*, using $\pi : \{1...k\} \rightarrow 2^{V_D}$, the descendents of each mother node. Only the rewriting of

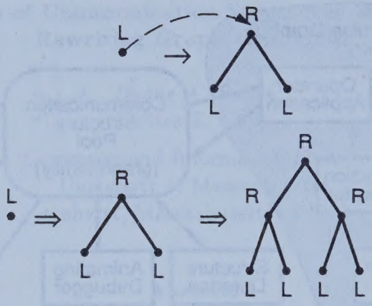
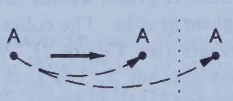


Figure 3: The grammar which generates binary trees (top), and the first several derivation steps (bottom).

the interface is affected by this decision:[1] whenever a edge between two nodes of an aggregate is rewritten, no copy of this edge is introduced between descendents which are members of different partitions.

An immediate advantage of AR graph grammars over other formalisms is the power of duplicating arbitrarily large aggregates. Envision, for example, the following production.[2]



Both daughter nodes are descendents of the mother node (as indicated by the dashed arrows), but are in different partitions (indicated by dotted lines); this production rewrites an arbitrarily large aggregate whose nodes are all labeled A to a graph twice its size (see Figure 4). The image nodes are instances of both daughter nodes, half stem from the left partition, and half from the right. This segregation requires there be no inherited edge between two groups, but at the same time, because of inheritance, each group maintains the topology of the rewritten aggregate.

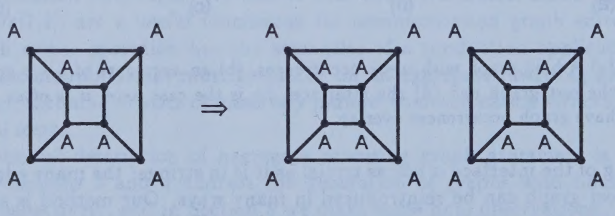


Figure 4: A single production can duplicate aggregates of arbitrarily large size. Here, we see the effect of the production $\overset{A}{\cdot} \rightarrow \overset{A}{\cdot}\;\overset{A}{\cdot}$, in which each daughter node is a descendent, but resides in its own partition.

---

[1] The trivial partitioning leads to the full crossbar.

[2] Throughout this paper, we indicate inheritance by dashed arrows, and partitioning using dotted lines.

In the following sections we provide theorems that show AR graph grammars with certain characteristics generate graphs with two useful properties common to massively parallel communication structures: bounded degree and symmetry.

3. Graphs with Bounded Degree

Communication structures for programs composed of thousands of processes are typically sparse. This is because the number of channels necessary for a particular process is often limited by some logically or physically manageable number. Furthermore, increasing the potential for communication at critical processes often degrades performance.

The following theorem suggests a condition that is satisfactory for bounding the degree of graphs generated by AR grammars.

Theorem 1 *An aggregate rewriting graph grammar generates graphs of bounded degree if each edge of a sentential form, and its descendents, can only be rewritten by a bounded number of production applications.* □

Since the degree of a graph increases only at nodes that have inherited edges, bounding the number of generations of nodes that inherit the same edge will bound the degree of generated graphs.

Another way of stating this constraint is that a node has a bounded number of descendents. Of course, if the set of productions for a grammar has a 'loop' in inheritance, a node with more than one immediate descendant participating in that loop could generate arbitrarily large numbers of nodes, each of whom inherits a copy of incident edges. For example, the node labeled C inherits edges from nodes labeled C in the production defining the cube grammar (Figure 5). Note, in particular that half the edges of any large cube are descendents of the single edge in the 1-cube. Thus we have

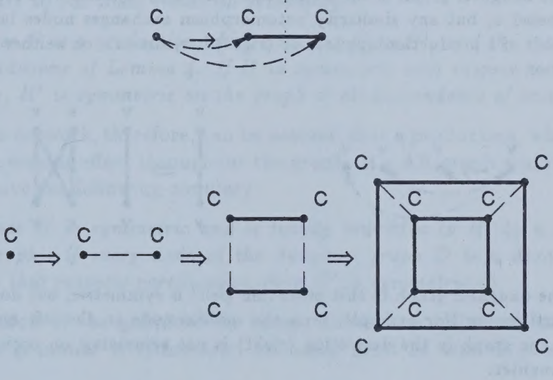


Figure 5: The grammar which generates binary cubes (top), and the first several derivation steps (bottom). Dashed edges indicate edges added as interface edges.

Corollary 2 *If there is no loop in the inheritance of nonterminals in an aggregate rewriting graph grammar, the grammar generates graphs of bounded degree.* □

The grammar for binary trees (see Figure 3) does not allow loops in the inheritance of edges (only R inherits from L), thus the degree of the tree is bounded. The grammar for the binary cube (see Figure 5) generates graphs of unbounded degree, and thus fails to meet the condition of this theorem (C can inherit from itself).

4. Graphs with Symmetry

In specifying code for individually programmed systems, it is often desirable to have the symmetry of similarly coded processes. Furthermore, where communication involves routing of messages, symmetry often is the key to efficient concurrent use of channels.

Many communication structures are not totally symmetrical, but are instead "symmetric" on some subset of logically related elements. More particularly, we have

Definition 3 *Two nodes u and v of a communication graph are* similar *if an automorphism of the graph maps u to v. A graph is node-symmetric (here, just* symmetric*) if every pair of nodes are similar. The graph H is* symmetric on subgraph *$S \subseteq H$ if every pair of points in S are similar, and the respective automorphisms are automorphisms of S.*

We say an automorphism α of a daughter graph respects partitioning*, if whenever u and v are in the same partition $\alpha(u)$ and $\alpha(v)$ are also. We say it* respects inheritance *if whenever u is a descendent of mother node m, so is $\alpha(u)$.* □

The necessity in respecting inheritance, and in respecting partitioning are demonstrated by Figures 6 and 7.

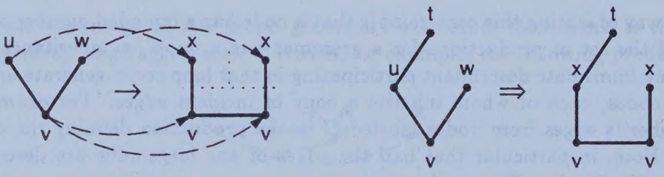


Figure 6: The daughter graph in the production (left) is symmetric with respect to nodes labeled v, but any similarity automorphism exchanges nodes labeled x. The result of a production application (right) is symmetric on neither nodes labeled x nor v.

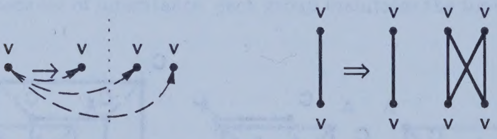


Figure 7: The daughter graph in this procution (left) is symmetric, but does not respect partitioning (for example, map the middle node to the left node), thus the image graph in the derivation (right) is not symmetric on occurrences of the daughter.

With these examples in mind, we turn to

Lemma 4 *Let $\mathcal{P}$ be a production which rewrites a graph H to H'. If the daughter graph D is symmetric on the the discrete subgraph D_v, composed of descendents of mother graph node v, and the similarity automorphisms respect inheritance and partitioning, then H' is symmetric on each image of D_v.* □

The symmetry of the daughter graph becomes symmetry in the image graph (see Figure 8); any pair of nodes which descend from the same aggregate node are similar.

Unfortunately, the previous lemma is insufficient to show that nodes are similar between occurrences of the daughter graph – that requires the host graph to exhibit symmetry. This

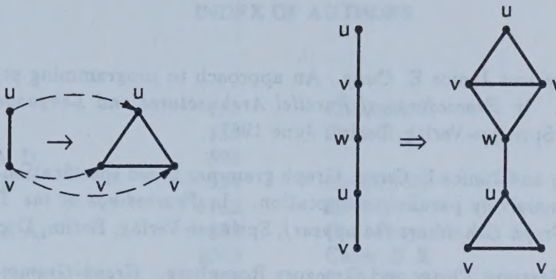


Figure 8: The symmetry of the daughter graph in the production (left) is preserved within each of its images in the resulting graph (right). However, there is no symmetry *between* daughter graph images.

would be extremely powerful, since it would allow the designer to specify homogeneous systems easily.

In order to assure *all* instances of descendents of a mother node are similar, we must impose some similarity between occurrences of the mother graph. The following theorem suggests that if the occurrences of the aggregate are "similar," the symmetry of daughter graph becomes symmetry in the transformed aggregate. We first define our notion of similarity of occurrences.

Definition 5 *A pair of occurrences M_i and M_j, $i, j \in J$ in a graph H are* similar *if there exists an automorphism α of J which maps i to j, and if α induces an automorphism of H which acts as $\psi_{\alpha(i)} \cdot \psi_i^{-1} : M_i \rightarrow M_{\alpha(i)}$. A graph is* symmetric with respect to the occurrences of M *if every pair of occurrences are similar.* □

This leads directly to our main result on symmetry.

Theorem 6 *Suppose P is a production which rewrites H to H', and the mother and daughter graphs meet the conditions of Lemma 4. If H is symmetric with respect to the occurrences of M, then for $v \in V_M$, H' is symmetric on the graph of all descendents of images of v.* □

The designer of a network, therefore, can be assured that a production, which rewrites a single occurrence has a similar effect throughout the graph. If a AR graph grammar's productions rewrite nodes, we have the following corollary.

Corollary 7 *Suppose H is symmetric and is totally rewritten to H' by a production with a singleton mother graph. If every node of the daughter graph D is a descendent, and D is symmetric in a way that respects partitioning, then H' is symmetric.* □

The sole production of the grammar for binary cubes meets these constraints. Since the start graph for this grammar is symmetric, the same must be true of every member of the language.

5. Discussion

AR graph grammars are extremely effective for specifying both families of graphs, and recursive structures. With proper restrictions, we expect to preserve many properties which are inherent to the structures associated with communication among thousands of processes.

Both a bound on degree and symmetry are important characteristics of massively parallel communication structures. When describing such structures with a graph editor, constraints on productions similar to those presented here can provide feedback to the programmer. Such metrics aid the designer of processes for highly parallel machines.

References

[1] Duane A. Bailey and Janice E. Cuny. An approach to programming process interconnection structures. In *Proceedings of Parallel Architectures and Languages Europe, vol. 2*, pages 112–123, Springer-Verlag, Berlin, June 1987.

[2] Duane A. Bailey and Janice E. Cuny. Graph grammar based specification of interconnection structures for massively parallel coumptation. In *Proceedings of the Third International Workshop on Graph Grammars (to appear)*, Springer-Verlag, Berlin, December 1986.

[3] Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg. *Graph-Grammars and Their Application to Computer Science and Biology. Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1979.

[4] Janice E. Cuny, Duane A. Bailey, John W. Hagerman, and Alfred A. Hough. *Simple Simon Programming Environment: A Status Report.* Appearing elsewhere in this volume.

[5] Frank Harary. *Graph Theory.* Addison-Wesley, Reading, Massachusetts, 1969.

INDEX OF AUTHORS

Aalo, V. 155
Aazhang, B. 135
Abbaszadeh, A. D. 598
Abrahamson, K. 624
Acar, L. 1162
Ait-Kheddache, A. 473
Akiyama, K. 1069
Albertos, P. 795
1202
Allebach, J. P. 483
Al-Muthairi, N. F. 558
Anantharam, V. 684
Antsaklis, P. J. 546
Arce, G. R. 513
753
Armoni, Y. 1081
Azadegan, F. 965
Aziz, G. 121
Babendreier, J. 560
Bai, S. 801
Bailey, D. A. 238
1234
Barmish, B. R. 117
Bar-Ness, Y. 928
Basu, S. 217
Be'ery, Y. 610
Bellow, M. E. 64
Berger, T. 919
Berman, N. 721
Bernhard, P. J. 1027
Bertossi, A. A. 46
Bhattacharya, P. P. 998
Bingulac, S. P. 558
Birkemeier, W. P. 453
Block, H. 662
Boles, W. W. 763
Bose, T. 192
Bouthellier, P. R. 1172
Bovik, A. C. 463
773
Bovopoulos, A. D. 979
Bredeson, J. G. 1103
Brown, D. P. 192
Bryant, R. L. 397
Casavant, T. L. 258
Cassandras, C. G. 287
Chang, S-M. 1040
Chiasson, J. 107
Choi, H-A. 4
268
Choi, M-R. 1142
Choi, S. 928
Choudhry, M. A. 562
Chow, B. S. 453
Ciliz, M. K. 203
Cobb, J. D. 544
Colgate, J. E. 821
Comeau, M. A. 634
Conn, H. E 898
Costello, Jr., D. J. 584
Csiszar, I. 917
Cuny, J. E. 238
1234
Dadoun, N. 624
Dandapani, R. 1059
Dasgupta, S. 1133
DeCarlo, R. A. 556
813
Dehne, F. 660
Delp, E. J. 731
775
DeMarco, C. L. 72
De Pano, N. A. 869
Desoer, C. A. 522
Dickinson, B. W 409
Douligeris, C. 989
Dunham, J. G. 967
Efstratiadis, S. N. 493
Emre, E. 534
Enab, Y-M. 839
Ephremides, A. 969
998
Esfahanian, A-H. 34
268
Esfandiari, F. 97
Farrell, J. A. 1123
Farvardin, N. 938
Fellows, M. R. 397
Ferreira, H. C. 596
Fiduccia, C. M. 201
Fine, T. L. 123
Fineberg, S. A. 258
Fischer, T. R. 955
Foias, C. 205

Foster, R. E. 753
Fowler, D. R. 889
Frazho, A. E. 205
Frederickson, G. 879
Garber, F. D. 622
Gburzynski, P. 908
Ge, Q. W. 305
Gelfand, S. 700
Gelfand, S. B. 731
Ghez, S. 1089
Ghosh, B. K. 1172
Gibson, J. D. 955
Gonzalez, O. R. 546
Gonzalez, T. F. 377
387
Goodrich, M. T. 849
Gudmundsson, G. 375
Gunther, O. 859
Gupta, G. 1050
Hached, M. 841
Haddad, A. H. 426
Hagerman, J. W. 238
Hakimi, S. L. 34
373
Hamdan, A. M. A. 560
Halverson, D. R. 163
Hansen, E. R. 54
Harrison, D. D. 938
Hassan, A. A. 600
Hedge, M. V. 564
Heegard, C. 946
Herro, M. A. 594
Ho, J. 121
Ho, Y-C. 279
Hogan, N. 821
Hopkins, M. A. 399
Hough, A. A. 238
Huffman, G. M. 622
Iftar, A. 1152
Ingram, M. A. 426
Iyengar, S. 1059
JaJa, J. 355
Javidi, B. 1142
Jha, N. K. 1050
Kabuli, M. G. 522
Kanefsky, M. 763
Kappos, E. 707
Karatzoglu, M. A. 969
Katsaggelos, A. K. 493
741
Ke, Y. 869
Keil, J. M. 889
Kent, R. E. 1204
Kerrick, D. D. 463
Khalil, H. K. 97
Khargonekar, P. P. 88
Kim, C. 1224
Kim, Y. K. 574
Kinnersley, N. G. 397
Kirkpatrick, D. G. 624
Knight, J. F. 335
Kogan, J. 1200
Kohli, R. 1101
Koo, B. 955
Krishna, H. 203
Krishnaprasad, S. 227
Krogh, B. H. 317
Kumar, S. P. R. 694
Kuo, S. M. 215
Kurki-Gowdara, S. 377
Lai, T. H. 1007
Lakshmanan, K. B. 634
Langston, M. A. 397
Lazar, A. A. 182
248
979
Lee, C. C. 574
Lee, G. 236
Lee, Y.-H. 743
Leung, J. Y-T. 650
Levy, H. 664
Li, J-H. 1113
Li, S. 409
Lindner, D. K. 560
Lodi, E. 46
Lovell, G. H. 829
Lu, C-C. 967
Lu, D. 125
Luccio, F. 46
Luh, J. Y. S. 839
Madani-Esfahani, S. M. 841
Mahajan, S. 14
Mahtre, K. 217
Manacher, G. K. 24
Mankus, T. A. 24
Martel, C. U. 1079
Masuda, S. 345
Mazumdar, R. 989
McHugh, J. 36
McLeish, M. 908
Mendel, J. M. 417
Meybodi, M. R. 1017

Meyer, G. G. L. 1190
Michel, A. N. 1113
1123
Milito, R. A. 664
Mingo, A. C. 1101
Mitter, S. K. 700
Mobed, M. 436
Modestino, J. W. 938
965
Monderer, B. 182
Morant, F. 795
Munro, J. I. 54
Murata, T. 270
Murray, J. J. 829
Naaman, L. 773
Naclerio, N. J. 345
Nakajima, K. 4
345
Narayan, P. 917
Nel, A. L. 596
Nielsen, P. A. 172
Ni, L. M. 268
Ntafos, S. 375
Onaga, K. 305
Ono, N. 305
Oommen, B. J. 54
O'Rourke, J. 869
898
Orsak, G. 135
Oruc, A. Y. 1037
Ozguner, U. 1152
1162
Pagli, L. 46
Panwar, S. R. 1081
Park, S. Y. 743
Passino, K. M. 335
Perl, Y. 36
Peters, J. G. 14
Petersen, I. R. 88
Peyravi, M. H. 1017
Pless, V. 620
Podrazik, L. J. 1190
Popoli, R. F. 417
Porod, W. 1113
1123
Pourboghrat, F. 811
Przytycka, T. 624
Rahgozar, M. 483
Rajala, S. A. 473
Rao, T. S. 513
Reddy, S. M. 1059
Rego, V. 66
1099
Riberio do Val, J. B. 713
Rim, C. S. 4
345
Robertazzi, T. G. 199
248
Rodger, S. 879
Rodriguez, M. A. 215
Rohlicek, J. R. 674
Root, W. L. 436
1181
Rosenkrantz, D. J. 1027
Rosenthal, A. 641
Rosier, L. E. 1214
Rouanne, M. 584
Rushforth, C. K. 598
Sadowsky, J. S. 145
Sadrolhefazi, A. 123
Sain, M. K. 524
Salam, F. M. A. 801
1142
Saluja, K. K. 1069
Scheuermann, P. 641
Schneider, S. 1037
Schwartz, J. T. 843
Schwartz, S. C. 199
1089
Schwederski, T. 258
Seo, J. H. 534
Serakos, D. 1181
Sharir, M. 843
Sherlekar, D. D. 355
Sherman, P. J. 205
Shi, P. 503
Shrivastava, Y. 1133
Shwartz, A. 721
Silva, M. 270
Simaan, M. 763
785
Sims, C. S. 562
Sira-Ramirez, H. 78
Song, J. 775
Sreenivas, R. S. 315
Stark, W. E. 564
600
Stojemnovic, I. 660
Strickland, S. G. 287
Sullivan, B. J. 741
Sung, Y. Y. 365
Szpankowski, W. 66
1099

Tai, H.-M. 534
Tan, H. L. 731
Tanida, T. 305
Telang, V. 594
Teneketzis, D. 564
Thistle, J. G. 327
Thomas, J. 172
Thomopoulos, S. C. A. 155
Thompson, M. W. 163
Thulasiraman, K. B. 634
Tornero, J. 1202
Towsley, D. 1081
Tuan, T-C. 373
Vakili, P. 279
Vanderwiele, J. M. 1103
VanLandingham, H. F. 399
Vastola, K. S. 1101
Vayada, T. P. 1079
Vardy, A. 610
Verdu, S. 1089
Viniotis, I. 998
Viswanathan, R. 155
Wang, Y. 1142
Ward, R. K. 503
Wessel, A. E. 446
Wey, C-L. 1040
Wicks, M. A. 556
Willsky, A. S. 674
Wise, G. L. 446
Wolf, J. K. 1081
Wong, E. 859
Wonham, W. M. 1
327
Wright, D. A. 596
Wyman, B. F. 524
Yang, T. H. 1007
Yao, K. 125
Yaz, E. 119
Yen, H.-C. 1214
Yeung, R. 919
Young, G. H. 650
Yousef, H. 785
Zak, S. H. 813
841
Zazanis, M. A. 297
Zheng, S-Q. 387
Zhou, K. 88
Zhu, Y. 409
Zribi, M. 107

Tai, H. M. 558
Tan, H. L. 731
Tanaka, T. 505
Tsang, N. 398
[illegible], D. 264
[illegible], J. G. 327
[illegible], L. 152
Thompson, S. G. A. 152
Thompson, M. W. 161
[illegible], C. B. 634
Turner, J. 1179
[illegible], M. 1081
[illegible] 328
[illegible], F. 479
[illegible], J. M. 1101
[illegible], H. F. 593
[illegible], R. S. 1131
[illegible], T. P. 1071
[illegible], R. 610
[illegible] 1029
[illegible] 306
[illegible], K. 133
Wang, Y. 1142
Ward, R. [illegible] 305
[illegible], A. F. 436
Wei, C. L. 1049
[illegible], A. 556
[illegible], A. M. 674
[illegible], C. L. 546
Wolf, J. R. 1081
Wong, E. 839
[illegible], W. M. 1
287
Wright, D. A.
[illegible], G. F. 534
Yang, T. H. 1007
Yao, K. 125
Yao, F. 119
[illegible], H. C. 1134
Young, [illegible] 919
Young, C. H. 650
[illegible], H. 789
[illegible], Y. H. 814
881
[illegible], M. A. 297
Zhang, S. Q. 367
[illegible] 35
[illegible], Y. 409
Zribi, M. 107

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1987:2
3 0112 008149236